BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(12) UK Patent Application (19) GB (11) 2 337 409 (13) A

(43) Date of Printing by UK Office 17.11.1999

(21) Application No 9918428.5

(22) Date of Filing 29.01.1998

(30) Priority Data
(31) 08795917 (32) 05.02.1997 (33) US

(86) International Application Data
PCT/US98/01729 En 29.01.1998

(87) International Publication Data
WO98/34350 En 06.08.1998

(51) INT CL[6]
H04Q 3/00

(52) UK CL (Edition Q)
H4K KDD KF42

(56) Documents Cited by ISA
US 5771386 A US 5754628 A US 5664008 A
US 5653559 A US 5450486 A

(58) Field of Search by ISA
INT CL[6] H04J 3/06
U.S. : 455/560; 455/439; 395/500

(71) Applicant(s)
Firsttel Systems Corporation
(Incorporated in USA - California)
100 Marine World parkway, Redwood Shores,
California 94065, United States of America

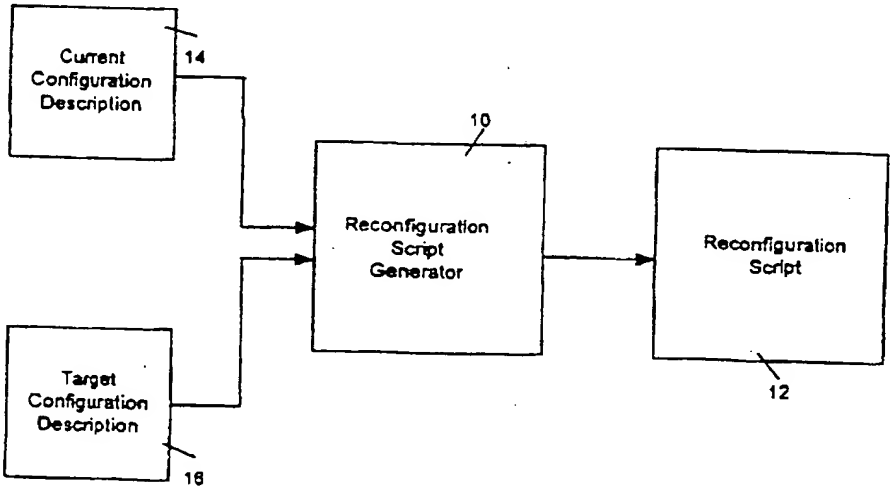
(72) Inventor(s)
Xiwen Ma
XiaoLin Tian
Steve Berriatua

(72) cont
Simon F Moloney
Mark A Russell

(74) Agent and/or Address for Service
Potts, Kerr & Co
15 Hamilton Square, BIRKENHEAD, Merseyside,
CH41 6BR, United Kingdom

(54) Abstract Title
Automatic generation of reconfiguration scripts for telecommunication devices

(57) An apparatus is programmed with a plurality of programming instructions for automatically generating a reconfiguration script comprising a plurality of configuration commands for reconfiguring a plurality of telecommunication devices of a telecommunication system, based on a current and a target descriptive image of the telecommunication system. The current descriptive image specifies the devices and their features included in the current configuration of the telecommunication system, whereas the target image specifies the devices and their features to be included in the target configuration of the communication system. The plurality of programming instructions generate the reconfiguration script employing feature dependency graph (FDG) data structures, a device model modeling the rules and behaviors of the telecommunication devices, and feature deletion/addition linklists.



GB 2 337 409 A

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04B

A2

(11) International Publication Number: WO 98/34350

(43) International Publication Date: 6 August 1998 (06.08.98)

(21) International Application Number: PCT/US98/01729

(22) International Filing Date: 29 January 1998 (29.01.98)

(30) Priority Data:
08/795,917 5 February 1997 (05.02.97) US

(71) Applicant: FIRSTTEL SYSTEMS CORPORATION [US/US]; 100 Marine World Parkway, Redwood Shores, CA 94065 (US).

(72) Inventors: MA, Xiwen; 1615 Chestnut Street, Berkeley, CA 94702 (US). TIAN, XiaoLin; 10192 Park Circle West #2, Cupertino, CA 95014 (US). BERRIATUA, Steve; 778 Olive Court, San Bruno, CA 94066 (US). MOLONEY, Simon; 150 D Street, Redwood City, CA 94063 (US). RUSSELL, Mark; 34 Sonoma Street, San Francisco, CA 94123 (US).

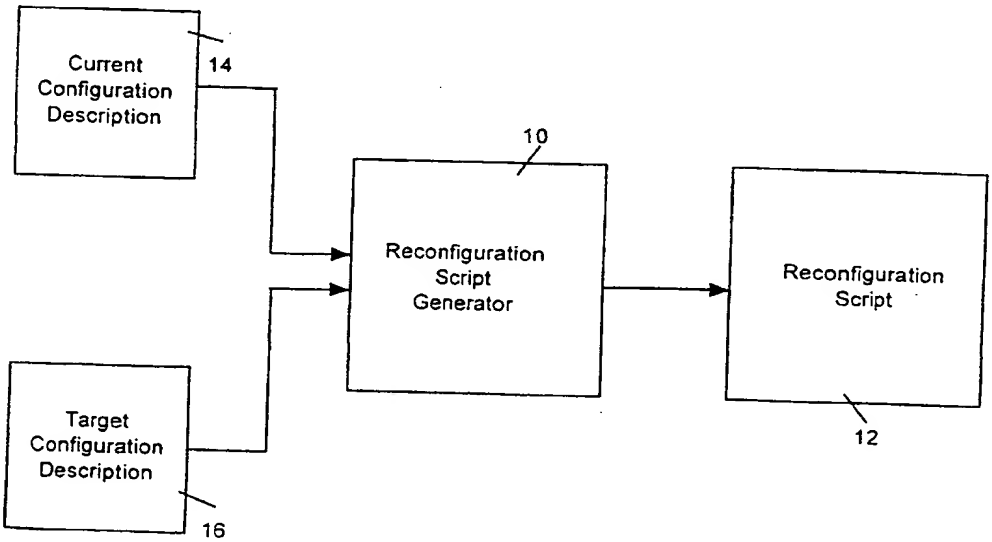
(74) Agents: AU YEUNG, Aloysius, T., C. et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 7th floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025 (US).

(81) Designated States: AL, AM, AT, AT (Utility model), AU (Petty patent), AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, CZ (Utility model), DE, DE (Utility model), DK, DK (Utility model), EE, EE (Utility model), ES, FI, FI (Utility model), GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SK (Utility model), SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published

Without international search report and to be republished upon receipt of that report.

(54) Title: AUTOMATIC GENERATION OF RECONFIGURATION SCRIPTS FOR TELECOMMUNICATION DEVICES



(57) Abstract

An apparatus is programmed with a plurality of programming instructions for automatically generating a reconfiguration script comprising a plurality of configuration commands for reconfiguring a plurality of telecommunication devices of a telecommunication system, based on a current and a target descriptive image of the telecommunication system. The current descriptive image specifies the devices and their features included in the current configuration of the telecommunication system, whereas the target image specifies the devices and their features to be included in the target configuration of the communication system. The plurality of programming instructions generate the reconfiguration script employing feature dependency graph (FDG) data structures, a device model modeling the rules and behaviors of the telecommunication devices, and feature deletion/addition linklists.

Automatic Generation Of Reconfiguration Scripts For Telecommunication Devices

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to the field of telecommunications. More specifically, the present invention relates to reconfiguration of a network of telecommunication devices.

2. Background Information

In the telecom industry, complex and specialized software and hardware systems are a given. Because each system is specialized, information used by one system is not easily manipulated for use in another. As a result, software designers are constantly asked to provide solutions, via software, that can integrate different systems in a consistent and easy-to-use manner.

Designing and building software that is consistently easy to use and can integrate and manipulate information from other systems is often extremely difficult. Device modeling gives to the software systems designer a method for representing behaviors of very complex devices in a model that is simple to use and understand. The model will remain the same even if the device changes; thus, the software designer is free to concentrate on the functional capability of a device.

Device emulation is used to mimic the behavior of a modeled device and its features. Where the device model is used in the context of device emulation, the device model is invoked as part of the application. Therefore, the

capability of an application can be expanded by simply providing a new device model for any given device type. Of course, different applications are free to use the emulation of a device type for different purposes. Essentially, device emulation provides the "how" to device modeling's "what". Device emulation allows new features to be introduced in the system in a standard way, thus avoiding code changes, which in turn leads a more reliable product, and shorter development cycle.

Device emulation enables system developers to focus more on the system's functional objectives than on the peculiarities of the device or network elements being managed. All information exchanges between the network management software and the network elements take the form: 1) what needs to be done--the action, and, 2) what to do it with--the data. As an intermediary between components, the device emulation adds interpretation or knowledge (the how) to the action-plus-data (the what). When a user makes a change to a device, the network management software interprets what that change means using device emulation and then makes that change directly on the device. The device's response is in turn interpreted so the management software can understand it.

In copending US Patent Application, no. 08/188,473, filed on Jan. 28, 1994, assigned to the assignee of the present invention, an improved method for efficiently modeling and emulating devices in a network of telecommunication systems was disclosed.

In addition to the complexity of these specialized software and hardware systems, constant changes, i.e. reconfiguration of devices in terms of their operating features and connectivity, is also a given. Thus, it is desirable to have an automated approach to reconfiguring telecommunication devices, in

particular, an approach that can leverage on the modeling and emulation technique disclosed in the copending application.

SUMMARY OF THE INVENTION

An apparatus is programmed with a plurality of programming instructions for automatically generating a reconfiguration script comprising a plurality of configuration commands for configuring/reconfiguring a plurality of telecommunication devices of a telecommunication system, based on a current and a target descriptive image of the telecommunication system. The current descriptive image specifies the devices and their features included in the current configuration of the telecommunication system, whereas the target descriptive image specifies the devices and their features to be included in the target configuration of the telecommunication system. The plurality of programming instructions generate the reconfiguration script employing feature dependency graphs (FDG) data structures, a device model modeling the rules and behaviors of the telecommunication devices, and feature deletion/addition linklists.

In one embodiment, the plurality of programming instructions implement a number of functions, including in particular, a first and a second function for generating a current and a target FDG data structure for the current and target configurations, based on the current and target descriptive images and a device model modeling the rules and behaviors of the telecommunication devices. The functions further include a third function for building a reconfiguration FDG data structure based on the current and target FDG data structures, a fourth function for generating a feature deletion and a feature addition linklist based on the reconfiguration FDG data structure, and a fifth function for generating the reconfiguration script comprising the configuration/reconfiguration commands for effectuating the desired reconfiguration.

BRIEF DESCRIPTION OF DRAWINGS

The present invention will be described by way of exemplary embodiments, but not limitations, illustrated in the accompanying drawings in which like references denote similar elements, and in which:

Figure 1 is a block diagram illustrating the present invention;

Figure 2 is a block diagram illustrating one embodiment of the reconfiguration script generator;

Figure 3 is a block diagram illustrating one embodiment of the current/target data structure;

Figure 4 is a flow diagram illustrating one embodiment of the method steps for loading data into the current/target data structure;

Figure 5 is a flow diagram illustrating one embodiment of the method steps for creating the current/target FDG data structure;

Figure 6 is a flow diagram illustrating one embodiment of the method steps for creating the reconfiguration FDG data structure;

Figure 7 is a block diagram illustrating one embodiment of the method steps for creating the deletion/addition linklists;

Figure 8 is a block diagram illustrating one embodiment of the method steps for generating the reconfiguration script;

Figure 9 is a block diagram illustrating an architecture for an integrated telecommunication network management system suitable for practicing the present invention;

Figure 10 is a block diagram illustrating a system view of a network of telecommunication devices incorporated with the integrated telecommunication network management system of **Fig. 9**;

Figure 11 is a block diagram illustrating selected portions of one embodiment of the network management and access layer, the voice element

management subsystem, and the voice element management databases of **Fig. 9** in further detail;

Figure 12 illustrates one example each of a current and a target image;

Figures 13 - 15 and **16 - 18** illustrate the current and the target FDG for the sample current image of **Fig. 12** respectively;

Figures 19 - 20 illustrate the reconfiguration FDG derived from the current and target FDGs of **Figs. 13 - 18**;

Figures 21 - 22 illustrate the deletion and addition linklists generated per the reconfiguration FDG of **Figs. 19 - 20**;

Figure 23 illustrate the reconfiguration script generated per the deletion and addition linklists of **Figs. 21 - 22**;

Figure 24 illustrates the concept and necessity for scope nodes; and

Figure 25 illustrated the concept of a kernel for the current and target FDGs.

DETAILED DESCRIPTION OF THE INVENTION

In the following description, various aspects of the present invention will be described. However, it will be apparent to those skilled in the art that the present invention may be practiced with only some or all aspects of the present invention. For purposes of explanation, specific numbers, materials and configurations are set forth in order to provide a thorough understanding of the present invention. However, it will also be apparent to one skilled in the art that the present invention may be practiced without the specific details.

Parts of the description will be presented in terms of operations performed by a computer system, using terms such as data, flags, bits, values, characters, strings, and/or numbers, consistent with the manner commonly employed by those skilled in the art to convey the substance of their work to others skilled in the art. As well understood by those skilled in the art, these quantities take the form of electrical, magnetic, or optical signals capable of being stored, transferred, combined, and otherwise manipulated through mechanical and electrical components of the computer system; and the term computer system include general purpose as well as special purpose data processing machines, systems, and the like, that are standalone, adjunct or embedded.

Various operations will be described as multiple discrete steps in turn in a manner that is most helpful in understanding the present invention, however, the order of description should not be construed as to imply that these operations are necessarily order dependent. In particular, these operations need not be performed in the order of their presentation.

Referring now to **Figure 1**, wherein a block diagram illustrating the present invention is shown. As illustrated, in accordance with the present invention, reconfiguration script generator **10** is provided to automatically

generate reconfiguration script **12** for telecommunication devices of a telecommunication system. Reconfiguration script **12** comprises of directly executable configuration commands for deleting/adding features of telecommunication devices, as well as deleting/adding the telecommunication devices (hereinafter, simply devices) from/to the telecommunication system.

Reconfiguration script **12** is generated based on current configuration descriptive image **14** and target configuration descriptive image **16**. Current configuration descriptive image **14** specifies the devices and their features included in the current configuration of the telecommunication system, whereas target configuration descriptive image **16** specifies the devices and their features included in the target configuration of the telecommunication system. Two examples, one each, for current and target configuration descriptive images **14** and **16** are illustrated in **Fig 12**. In exemplary current and target configuration descriptive images **14** and **16**, station 4 is being changed from being the pilot of a directory number hunt (DNH) group to just a member of the DNH group, while station 5 is being changed from a member of the DNH group to being the pilot of the DNH group. Additionally, station 7 is being deleted, while station 8 is being added.

As will be described in more detail below, reconfiguration script generator **10** generates reconfiguration script **12** employing FDG data structures, a device model modeling the telecommunication devices, and deletion/addition linklists. While for ease of explanation, current and target descriptive images **14** and **16** have been described as two separate elements, as will be appreciated by those skilled in the art, the present invention may be practiced with current and target descriptive images **14** and **16** being provided to reconfiguration script generator **10** as two separate descriptive images or as a single combined descriptive image.

Referring now to **Figure 2**, wherein a block diagram illustrating one embodiment of reconfiguration script generator **12** is shown. For the illustrated embodiment, reconfiguration script generator **12** comprises a plurality of objects[1], including root **18**, current and target configurations **20** and **22**, current and target FDGs **24** and **26**, reconfiguration FDG **28**, reconfiguration linklists **29**, and reconfiguration script **30**. For the illustrated embodiment, root **18** includes Set_Data, Build_CT_FDG, Build_Rec_FDG, Enumerate and Output methods, and owns objects **20 - 30**. Current and target configurations **20** and **22** hold the current target and configuration data, provided by current and target configuration descriptive images **14** and **16** respectively. Current and target configurations **20** and **22** are created using Set_Data. Current and target FDGs **24** and **26** hold the current and target configuration data complemented with feature scope and dependency data for representing the feature dependencies of the current and target telecommunication devices, i.e., telecommunication devices of the current and target configurations, in accordance with graph theory. Current and target FDGs **24** and **26** are created using Build_CT_FDG. Reconfiguration FDG **28** hold the reconfiguration data complemented with feature scope and dependency data for representing the feature dependencies of the impacted features of the impacted telecommunication devices, in accordance with graph theory. Reconfiguration FDG **28** is created using Build_Rec_FDG. Reconfiguration linklists **29** are ordered lists of the deletion and addition actions that need to be taken to reconfigure the telecommunication system from the current configuration to the desired target configuration. Reconfiguration linklists **29** are created using Enumerate. Reconfiguration script **30** contains the executable configuration commands to reconfigure the telecommunication system from the current configuration to the desired target configuration. Reconfiguration script **30** is created using Output.

[1] The term "object" is used herein to mean a software entity that includes data and methods that operate on the data, as it is understood by those skilled in the art of object oriented programming.

Referring now to **Figure 3**, wherein a block diagram illustrating one embodiment of a data structure suitable for implementing current/target configuration **20** or **22** is shown. For the illustrated embodiment, data structure **32** includes name field **34** for identifying whether data structure **32** holds current or target configuration data. Data structure **32** further includes station field **36**, where each instance holds the basic data, such as device or set type, identification, and so forth, for a telecommunication device, such as a telephone set, also referred to as a telephone station. Data structure **32** further includes button field **38**, where each instance holds the button number, and its feature list for a button instance. Data structure **32** further includes feature field **40**, where each instance holds the feature identification, and its parameter list for a feature instance. Lastly, data structure **32** further includes parameter field **42**, where each instance holds the parameter identification, and its parameter value for a parameter instance.

Figure 4 illustrates the operational flow for one embodiment of Set_Data. As shown, for the illustrated embodiment, upon invocation, Set_Data reads current/target configuration descriptive image **14** or **16**, and locates a station. Upon locating a station, Set_Data sets a station instance with its basic data, as provided by current/target configuration descriptive image **14** or **16**, step **102**. Next, Set_Data locates a button of the station. Upon locating a button, Set_Data appends the button's identification to the button list of the station, and create a button instance, step **104**. Set_Data then locates a feature of the button. Upon locating a feature, Set_Data appends the feature's identification to the feature list of the button, and create a feature instance, step **106**. Set_Data then locates a parameter of the feature. Upon locating a parameter, Set_Data append the parameter's identification to the parameter list of the feature, and create a

parameter instance, step **108**. Set_Data then locates the parameter's value, and set the parameter instance with the located parameter value, step **110**.

Set_Data then looks for more parameters of the feature, step **112**, and repeats steps **108 - 112** until all parameter instances for the feature have been created and their values set. Set_Data then looks for more features of the button, step **114**, and repeats steps **106 - 114** until all feature instances for the button have been created and their parameter values for all parameters set. Set_Data then looks for more buttons of the station, step **116**, and repeats steps **104 - 116** until all button instances for the station have been created, and their feature as well as parameter instances created, and values set. Set_Data then looks for more stations, step **118**, and repeats steps **102 - 118**, i.e. the entire process, until all station instances have been created, including all button, feature, and parameter instances, and parameter values.

Figure 5 illustrates the operational flow for one embodiment of Build_CT_FDG. As shown, for the illustrated embodiment, upon invocation, Build_CT_FDG reads data held in current/target configuration **20** or **22**, and for each button/feature, Build_CT_FDG creates a node, step **120**. Build_CT_FDG then retrieve feature dependency relationships for all buttons and features from a device model modeling the rules and behaviors of the telecommunication devices, step **122**. Then, for each required feature scope to enforce feature dependency, Build_CT_FDG also creates a node, step **124**. Lastly, for each feature dependency relationship, Build_CT_FDG creates an arc, step **126**.

An exemplary approach for modeling the rules and behaviors of the telecommunication devices is disclosed in the above identified copending US

Patent Application. **Figure 24** illustrates the concept and the necessity for feature scopes. At the left, two features of two buttons (represented as nodes **502 - 508**), including their feature dependencies (represented as arcs **514**) are illustrated. Employing a left-to-right depth first approach, the order of removal of the four button features are as shown, i.e. button 1/feature 2, button 2/feature 4, button 2/feature 3, and button 1/feature 1. However, if the graph is altered with the addition of the private and public scope nodes **510** and **512**, and feature dependency arcs **516**, the order of removal, using the same traversal technique, would be as shown, button 2/feature 4, public, button 1/feature 2, private, button 2/feature 3, and button 1/feature 1. Note that the order of removal for button1/feature 2 and button 2/feature 4 have been swapped. As appreciated by those skilled in the art, while sometimes telecommunication device features do not have explicit feature dependencies on each other, they nevertheless have certain order requirement between them. For example, while the feature 3-way conferencing (3WC) is not exactly "feature dependent" on telephone number, nevertheless 3WC must be removed prior to removing a telephone number assigned to a set. Thus, to maintain the order, the present invention employs the above described feature scopes. Two examples, one each, of the current and target FDGs are illustrated in **Figures 13 - 15** and **Figures 16 - 18** respectively. The two exemplary FDGs are current and target FDGs for the exemplary current and target configuration image of **Figure 12**. Note that the current and target FDGs presented are for illustration purposes only. For the illustrated embodiment, reconfiguration script generator **10** maintains the data for these FDGs in FDG data structures, but does not actually render the FDGs, as the actual rendering are unnecessary for the automated process of the present invention. Of course, for alternative embodiments, the actual rendering may be performed for usability or other purposes.

Referring now to **Figure 6**, wherein a flow diagram illustrating one embodiment of the method steps of Build_Rec_FDG is shown. For the

illustrated embodiment, upon being invoked, Build_Rec_FDG compares the current and target FDGs, and identifies a kernel section of the two FDGs that require no configuration action, i.e. neither deletion nor addition of a feature, step **128**. The kernel section of the two FDGs is the section comprising "unimpacted" feature nodes emanating from the root node. An "unimpacted" feature node is a node not being modified nor dependent on a node being modified. The kernel section may be identified using any one of a number of graph comparison techniques. Skipping now to **Figure 25**, wherein a block diagram illustrating the concept of kernel is shown. Illustrated at the top half of the figure are two sample current and target FDGs **516** and **518**. Illustrated at the lower half of the figure is a sample reconfigure FDG **520**. Current FDG **516** includes nodes a through e, whereas target FDG **518** includes nodes a, b', c, e and f. In other words, b is being modified; d is being deleted; and f is being added. To modify b, e has to removed and added back later, by virtue of its dependency on b. Therefore, a and c (emanating from a) are the only nodes "unimpacted". Thus, kernel section **522** comprises of nodes a and c.

Returning now to **Figure 6**, once the kernel section of the current and target FDGs **24** and **26** have been identified, Build_Rec_FDG creates the "delete" nodes (annotated with the "-" sign in subsequent figures) for reconfigure FDG **28** in view of current FDG **24**, step **130**. Recall from the earlier discussion, these "delete" nodes are the nodes of current FDG **24** that are "outside" of the kernel, and includes those that are temporarily deleted to allow changes to be made to their predecessor nodes. Next, Build_Rec_FDG creates the "addition" nodes (annotated with the "+" sign in the subsequent figures) for reconfigure FDG **28** in view of target FDG **26**, step **132**. Similarly, "addition" nodes are the nodes of target FDG **26** that are "outside" of the kernel. However, nodes that already exist as "delete" nodes are not duplicated, but reannotated as "change" nodes (annotated with both the "+" and "-" signs in subsequent figures).

An exemplary reconfiguration FDG is illustrated in **Figures 19 - 20**. The exemplary reconfiguration FDG is build by Build_Rec_FDG, based on the exemplary current and target FDGs of **Figures 13 - 15** and **Figures 16 - 18**. Similarly, note that the reconfiguration FDG is presented for illustration purpose only. For the illustrated embodiment, reconfiguration script generator **10** maintains the data for the reconfiguration FDGs in a FDG data structure, but does not actually render the FDG, as the actual rendering are unnecessary for the automated process of the present invention. Of course, for alternative embodiments, the actual rendering may be performed for usability or other purposes.

Referring now to **Figure 7**, wherein a flow diagram illustrating the method steps of one embodiment of Enumerate is shown. For the illustrated embodiment, Enumerate traverses reconfiguration FDG **28**, first generating the entries for the "deletion" linklist, then the entries for the "addition" linklist. Two examples, one each, for the "deletion" and "addition" linklists are shown in **Figures 21** and **22**. These are linklists generated by Enumerate based on the exemplary reconfiguration FDG of **Figures 19** and **20**.

Upon being invoked, Enumerate starts with the root node, and selects a dependency arc and the dependent node at the end of the arc (also referred to as the "child" node), steps **134** and **136**. Enumerate determines if the "child" node has descendant nodes, step **138**. Enumerate repeats steps **134 - 138** until a "child" node without descendant node is reached (also referred to as a "leaf" node). Upon locating the "leaf" node, Enumerate determines if the node is to be deleted, step **140**. If so, Enumerate adds the node to the "delete" linklist, step **142**, otherwise Enumerate proceeds directly to step **144**.

At step **144**, Enumerate backtracks to the "parent" node. Enumerate determines if the "parent" node has more dependency arcs, step **146**.

If so, Enumerate returns to step **134** and proceeds as described earlier. Otherwise, Enumerate determines if the "parent" node is the root node, step **148**. If the "parent" node is not the root node, Enumerate returns to step **140** and proceeds as described, otherwise Enumerate proceeds to perform steps **150 - 164**.

Steps **150 - 164** are similar to steps **134 - 148** except Enumerate is looking for nodes to be added, and generating entries for the "addition" linklist.

While the present invention has been described with the employment of separate "deletion" and "addition" linklists, with the "deletion" linklist being generated first, as will be appreciated by those skilled in the art, the number of linklists and their order of generation are merely for illustrative purpose only. The present invention may be practiced with the "addition" linklist being generated first, and/or using only a single combined linklist.

Referring now to **Figure 8**, wherein a flow diagram illustrating the method steps of one embodiment of Output is shown. For the illustrated embodiment, Output first generates the executable configuration commands for the deletion actions For the illustrated embodiment, non-executable configuration commands are also generated as "commentary" for the feature scope nodes that are to be "deleted" and "added". Recall, the feature scopes are not "real" features of the telecommunication devices. They are merely inserted in the FDGs to enforce certain implicit dependency between the features. An exemplary reconfiguration script having a number executable configuration commands (as well as non-executable pseudo configuration commands) is shown in **Figure 23**. These executable/nonexecutable (pseudo) configuration commands are generated by Output based on the exemplary linklists of **Figures 21** and **22**.

Referring now to **Figure 9**, wherein a block diagram illustrating an architecture of an integrated telecommunication network management system suitable for practicing the present invention is shown. The network elements, such as voice elements **212a**, data elements **212b**, and other elements **212c**, are accessed through a network management and access layer **214**. Particular examples of voice, data, and other elements **212a - 212c** include telephone services, provisioning, adding or deleting telephone options, trunk lines, provisions of bandwidth for Wide Area Network applications, and service control points. The network management and access layer **214** implements the specific device protocols, thereby freeing the core of the management software from such device details. The software core is typically organized into a number of subsystems **216a - 216e**, with each subsystem dedicated to managing a "family" of network elements, such as voice elements, data elements, network alarms, and Station Message Detail Recording (SMDR) collections. Typically, each subsystem also has its own databases **218a - 218e** for storing various profile, operational, and management data. In particular, these data include device type models where the various network devices being managed are modeled. The various management subsystems **216a - 216e** use the model data to emulate network devices when interpreting user or system actions. Additionally, the data are accessed by various management applications **222a - 222e**, such as billing, inventory, cable records, and service orders, through a database interface such as a Structured Query Language (SQL) interface.

There are many variations on how these elements **212a - 212c**, **214**, **216a - 216e**, **218a - 218e**, **220**, and **222a - 222e** are architecturally arranged. In some variations, the functions provided by some of these elements are combined, while in other variations, the functions provided by these elements may be further subdivided. The architecture illustrated is intended to be representative of a broad category of integrated telecommunication network management systems where device modeling and emulation is employed for performance and extendibility.

Figure 10, a block diagram, illustrates an exemplary telecommunication network system incorporating the integrated network management system of **Fig. 9**. The exemplary telecommunication network system **224** is comprised of a number of voice elements **212a**, a number of data elements **212b**, and a number of other elements **212c**, connected to each other through a network **226**. Particular examples of suitable networks include telephone company deployed voice and data service networks for business customers or hybrid networks that are built by large end-users of telecommunication services. Additionally, the exemplary network **224** further comprises a number of servers **228** executing the network management and access layer and the back end portions of the management subsystems and applications, and storing some portions of the system databases described earlier. Particular examples of suitable servers include workstations with non-proprietary operating systems like UNIX or proprietary network operating systems, that execute file retrieval and database record retrieval processes. In one embodiment, server **228** includes processor **229a**, memory **229b**, storage **229c** and network interface **229d** coupled to each other as shown. These elements perform their conventional function of processing, storage, interfacing, and so forth. Furthermore, the exemplary network **224** comprises a number of clients **228** executing the front end portions of the subsystem and applications, and storing the remaining portions of the system databases described earlier. Particular examples of suitable clients include DOS and Windows client applications or any program executing on a workstation accessing or storing information on the servers. Network management users manage the network through the clients **230**, which accomplish the management functions in cooperation with the servers **228**.

While a distributed telecommunication network system is being illustrated as being suitable for practicing the present invention, as will be

appreciated by those skill in the art, the present invention may also be practiced with a centralized telecommunication network system.

Figure 11, a block diagram, illustrates the relevant portions of one embodiment of the network management and access layer, the voice element management subsystem, and the voice element management databases of **Figure 9**. For the illustrated embodiment, the voice element management databases comprise a screen database **232a**, a station configuration database **232b**, a feature and parameter database **232c**, a switch command and sequence database **232d**, a request database **232e**, and a communication protocol database **232f**, located on the client and server as shown. The screen database stores user interface screens. The station configuration database **232b** further stores rules and behaviors of device types—the device models—whereas the feature and parameter database **232c** further stores the operational data of the devices. In the presently preferred embodiment, the device type models are copied into memory during operation to allow faster access and traversal. Rules and behaviors of device types, device type models, operational data of devices, and their usages will be described in further detail below.

The voice element subsystem comprises a user interface **234a**, a rules processor **234b**, a switch translator **234c**, and a request/response processor **234d** executed on the client and server as illustrated. These elements **234a - 234d** access the various databases **232a - 232e** as shown. To request a switch to perform an operation, a user interacts with the user interface **234a** to generate requests. T he requests are validated by the rules processors **234b**. The validated requests are in turn translated into process requests by the switch translator **234c**. The process requests are then transformed into switch command requests by the request/response processor **234d**. Response requests from the switch are transformed back to process requests by the request/response processor **234d**. The process requests are in turn translated back to validated requests by the switch

translator **234c**. The validated requests are then forwarded from the rules processor **34b** to the user through the user interface **234a**.

Additionally, the switch translator **234c** comprises a compiler for performing the above described compilations of the rules and behaviors of device types into device type models, whereas the rules processor **234b** comprises an array of evaluation functions for performing the above described evaluations of the model predicates of the device type models against the operational data of the devices while traversing the device type models. The compiler and the evaluation functions will be described in further detail below.

Lastly, the network management and access layer comprises a communication protocol processor **234e** executed on a server. The communication protocol processor **234e** is used to transfer switch commands and responses to and from the switches **236**.

While for ease of understanding, the present invention has been described with a particular embodiment of the voice element subsystem and its databases, based on the above descriptions, it will be appreciated that the present invention may be practiced without many of these details. Furthermore, the present invention may be practiced with other network element management subsystems.

While the method and apparatus of the present invention have been described in terms of the above illustrated embodiments, those skilled in the art will recognize that the invention is not limited to the embodiments described. The present invention can be practiced with modification and alteration within the spirit and scope of the appended claims. The description is thus to be regarded as illustrative instead of restrictive on the present invention.

Thus, a method and an apparatus for automatically generating reconfiguration scripts for telecommunication devices have been described.

CLAIMS

What is claimed is:

1. An apparatus comprising

(a) a storage medium coupled to the execution unit and having stored therein a plurality of programming instructions for implementing a reconfiguration script generator for generating a reconfiguration script having a plurality of executable configuration commands for configuring/reconfiguring a plurality of telecommunication devices, based on a current and a target configuration descriptive image of telecommunication devices; and

(b) an execution unit coupled to the storage medium for executing the plurality of programming instructions.

2. The apparatus as set forth in Claim 1, wherein the reconfiguration script generator includes a first function for initializing a current configuration data structure with current configuration data based on the current configuration descriptive image of telecommunication devices.

3. The apparatus as set forth in Claim 1, wherein the reconfiguration script generator includes a first function for initializing a target configuration data structure with target configuration data based on the target configuration descriptive image of telecommunication devices.

4. The apparatus as set forth in Claim 1, wherein the reconfiguration script generator generates the reconfiguration script employing feature dependency graphs to account for dependencies between features of telecommunication devices.

5. The apparatus as set forth in Claim 4, wherein the reconfiguration script generator includes a first function for building a current configuration feature dependency graph data structure to account for dependencies between current features of current telecommunication devices.

6. The apparatus as set forth in Claim 5, wherein the reconfiguration script generator further includes

a second function for building a target configuration feature dependency graph data structure to account for dependencies between target features of target telecommunication devices, and

a third function building a reconfiguration feature dependency graph data structure based on the current and target configuration feature dependency graph data structures to account for dependency between impacted features of impacted telecommunication devices.

7. The apparatus as set forth in Claim 4, wherein the reconfiguration script generator includes a first function for building a target configuration feature dependency graph data structure to account for dependencies between target features of target telecommunication devices.

8. The apparatus as set forth in Claim 1, wherein the reconfiguration script generator generates the reconfiguration script employing rules and behaviors of the telecommunication devices modeled by a device model to determine dependencies between features of telecommunication devices.

9. The apparatus as set forth in Claim 1, wherein the reconfiguration script generator generates the reconfiguration script employing at least one ordered linklist to enumerate deletion and addition operations required to configure/reconfigure the telecommunication devices.

10. The apparatus as set forth in Claim 9, wherein the reconfiguration script generator includes a first function for enumerating a number of deletion operations required to configure/reconfigure the telecommunication devices in an ordered linklist.

11. The apparatus as set forth in Claim 10, wherein the first function further enumerates a number of addition operations required to configure/reconfigure the telecommunication devices in the ordered linklist.

12. The apparatus as set forth in Claim 9, wherein the reconfiguration script generator includes a first function for enumerating a number of addition operations required to configure/reconfigure the telecommunication devices in an ordered linklist.

13. The apparatus as set forth in Claim 1, wherein the reconfiguration script generator includes a first function for outputting the plurality of executable configuration commands based on at least one ordered linklist identifying deletion and addition operations required to configure/reconfigure the telecommunication devices.

14. A machine implemented method comprising the steps of:

a) receiving a current configuration descriptive image of telecommunication devices;

b) receiving a target configuration descriptive image of telecommunication devices; and

c) automatically generating a reconfiguration script having a plurality of executable configuration commands for configuring/reconfiguring a plurality of telecommunication devices, based on the received current and target configuration descriptive images of telecommunication devices.

15. The method as set forth in Claim 14, wherein step (c) includes initialization of a current configuration data structure with current configuration data based on the current configuration descriptive image of telecommunication devices.

16. The method as set forth in Claim 14, wherein step (c) includes initialization of a target configuration data structure with target configuration data based on the target configuration descriptive image of telecommunication devices.

17. The method as set forth in Claim 14, wherein step (c) includes employment of feature dependency graph data structures to account for dependencies between features of telecommunication devices.

18. The method as set forth in Claim 17, wherein step (c) includes building of a current configuration feature dependency graph data structure to account for dependencies between current features of current telecommunication devices.

19. The method as set forth in Claim 18, wherein step (c) further includes building of a target configuration feature dependency graph data structure to account for dependencies between target features of the telecommunication devices, and

building of a reconfiguration feature dependency graph data structure based on the current and target configuration feature dependency graph data structures to account for dependency between impacted features of impacted telecommunication devices.

20. The method as set forth in Claim 17, wherein step (c) includes building of a target configuration feature dependency graph data structure to account for dependencies between target features of target telecommunication devices.

21. The method as set forth in Claim 14, wherein step (c) includes employment of rules and behaviors of the telecommunication devices modeled by a device model to determine dependencies between features of the telecommunication devices.

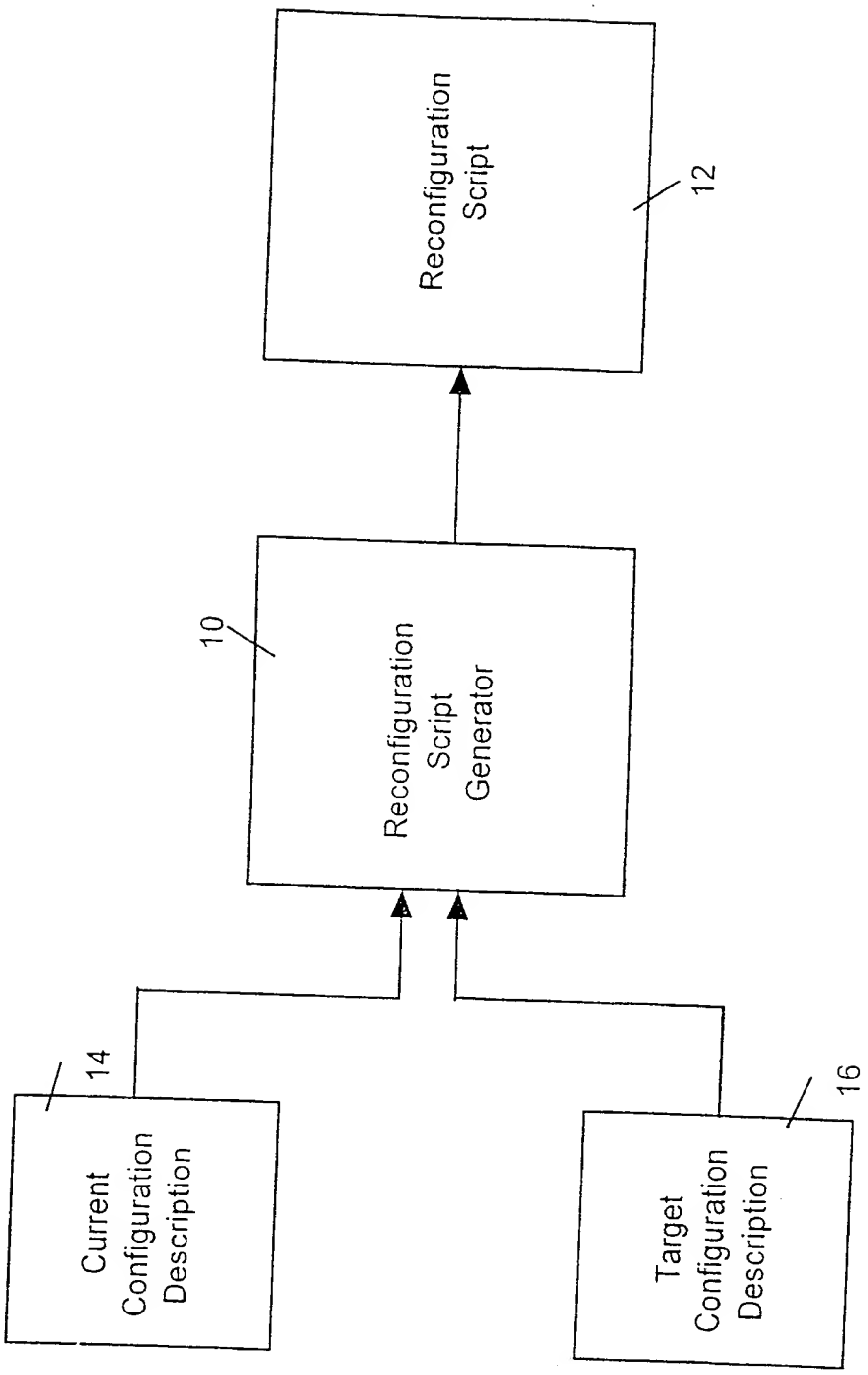
22. The method as set forth in Claim 14, wherein step (c) includes employment of at least one ordered linklist to enumerate deletion and addition operations required to configure/reconfigure the telecommunication devices.

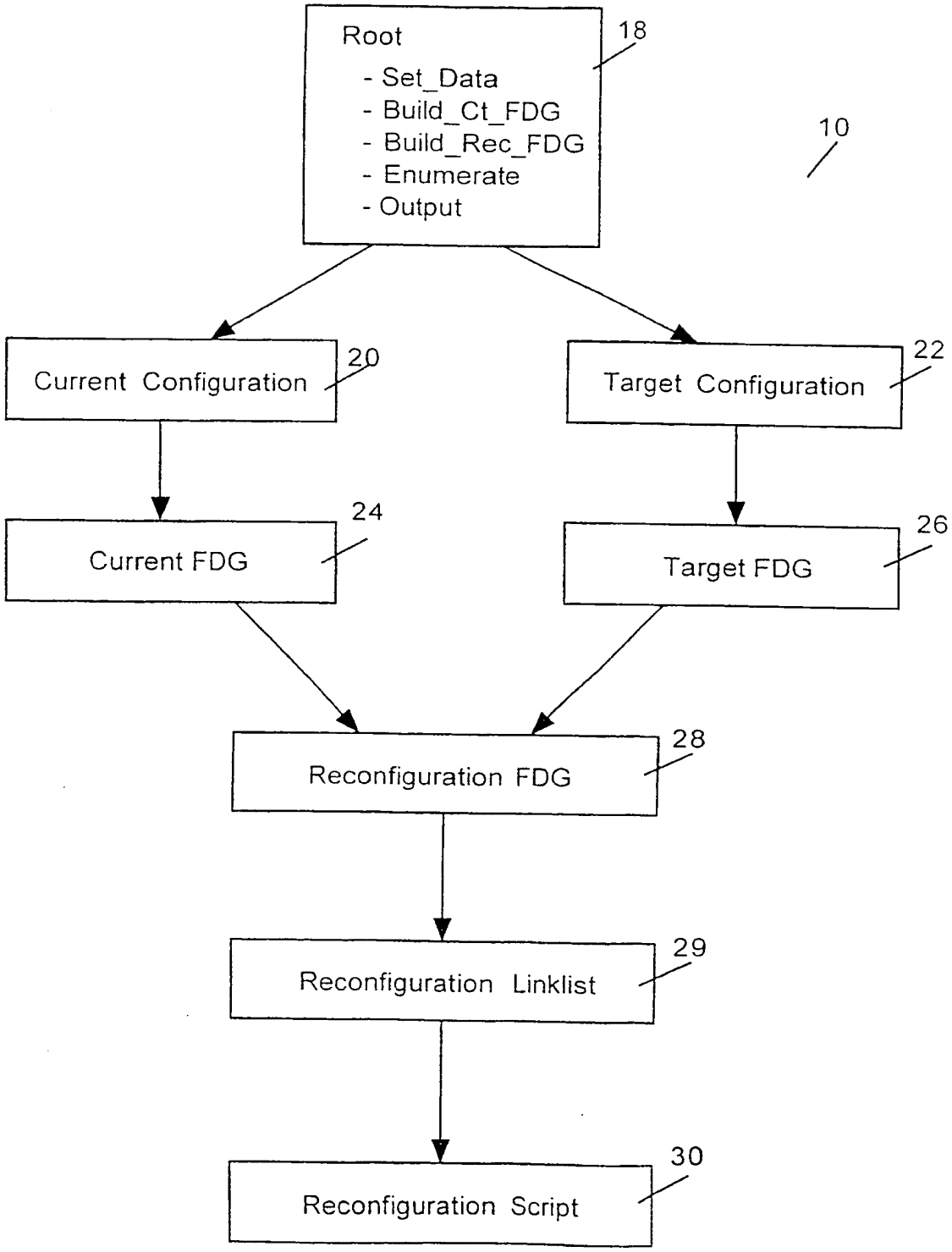
23. The method as set forth in Claim 22, wherein step (c) includes enumeration of a number of deletion operations required to reconfigure the telecommunication devices in an ordered linklist.

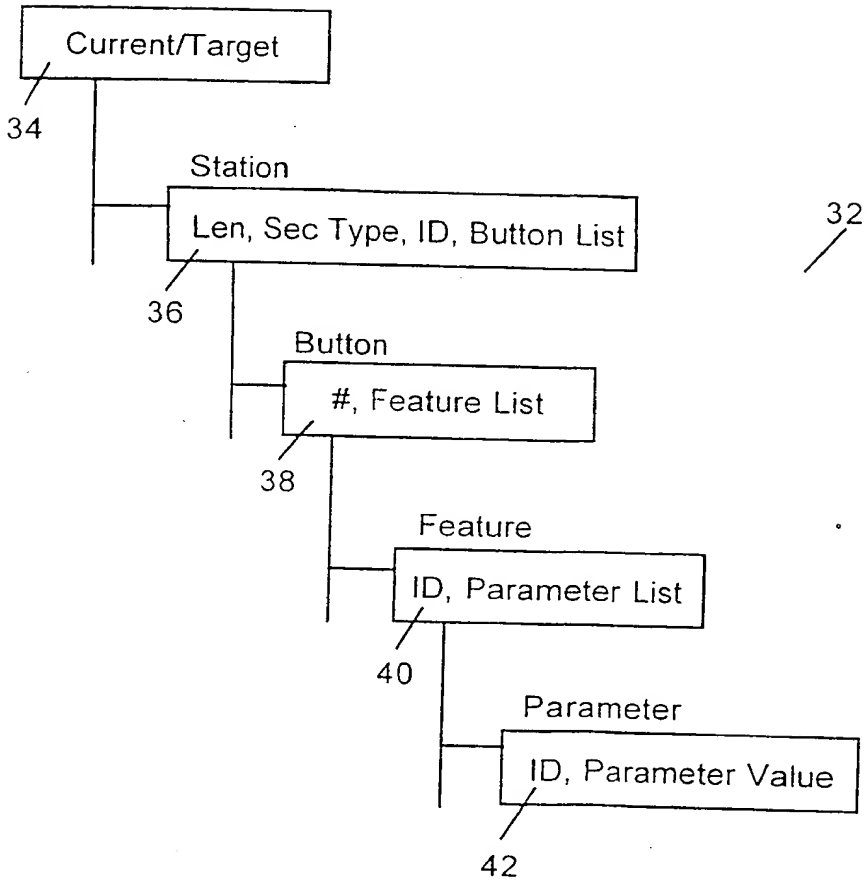
24. The method as set forth in Claim 23, wherein step (c) further includes enumeration of a number of addition operations required to configure/reconfigure the telecommunication devices in the ordered linklist.

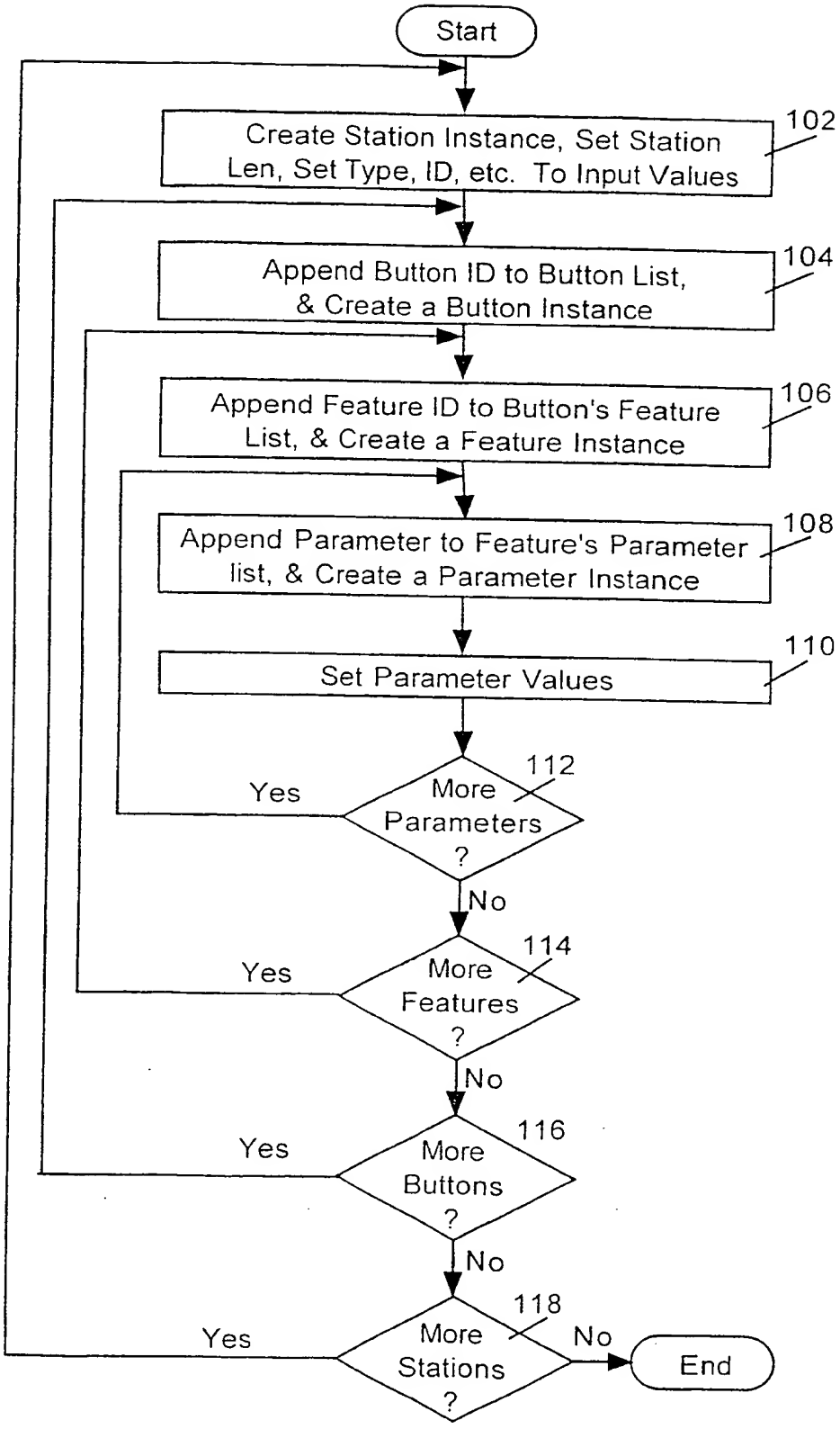
25. The method as set forth in Claim 22, wherein step (c) includes enumeration of a number of addition operations required to configure/reconfigure the telecommunication devices in an ordered linklist.

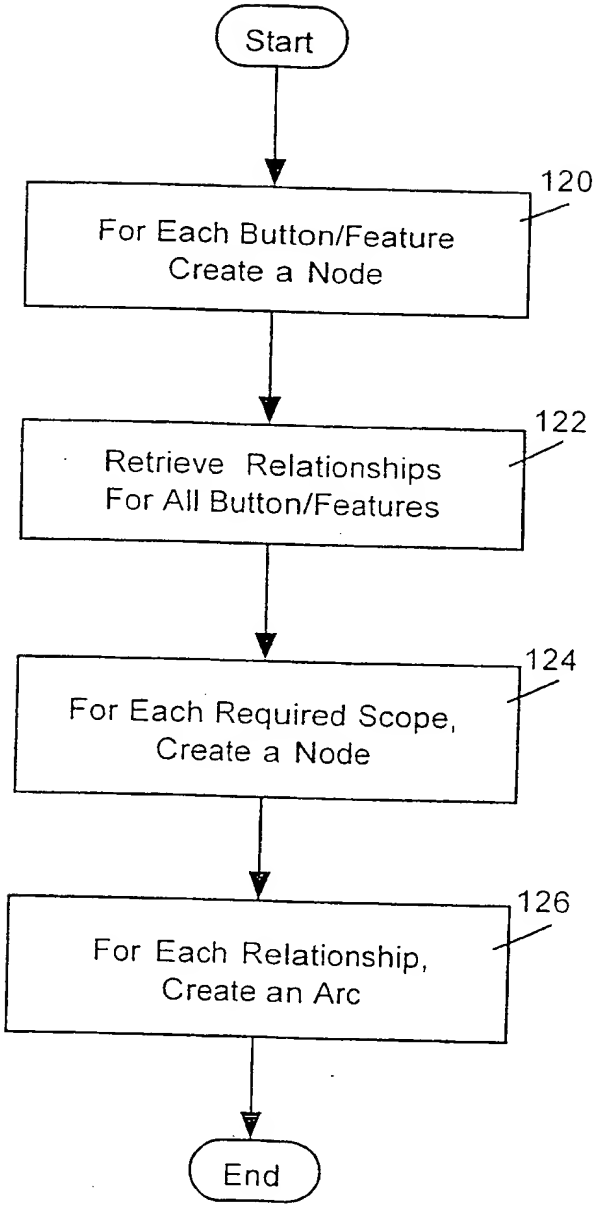
26. The method as set forth in Claim 14, wherein step (c) includes output of the plurality of executable configuration commands based on at least one ordered linklist identifying deletion and addition operations required to configure/reconfigure the telecommunication devices.

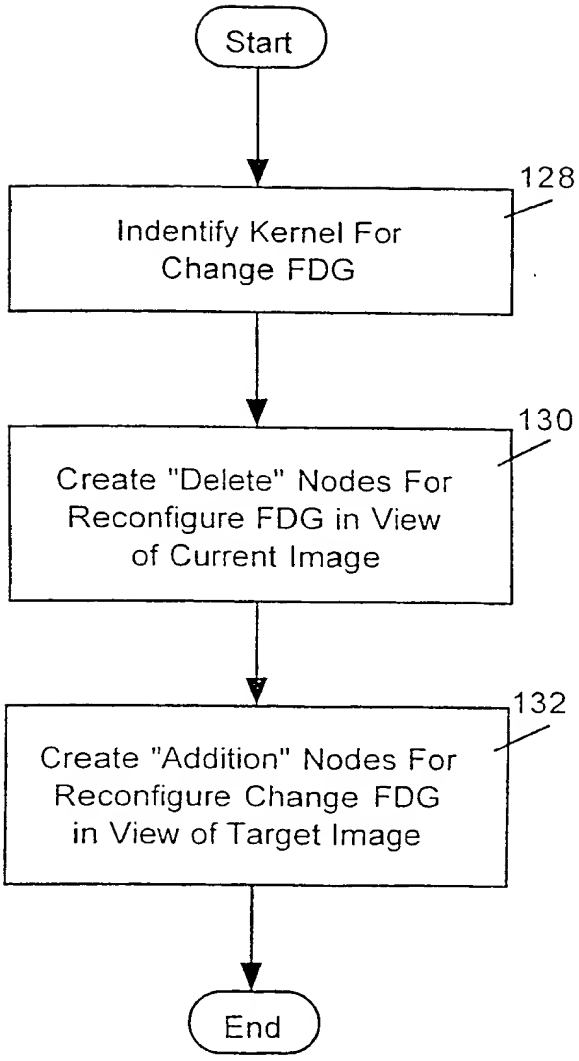

Reconfiguration Script
12
Reconfiguration Script Generator
10
Current Configuration Description
14
Target Configuration Description
16

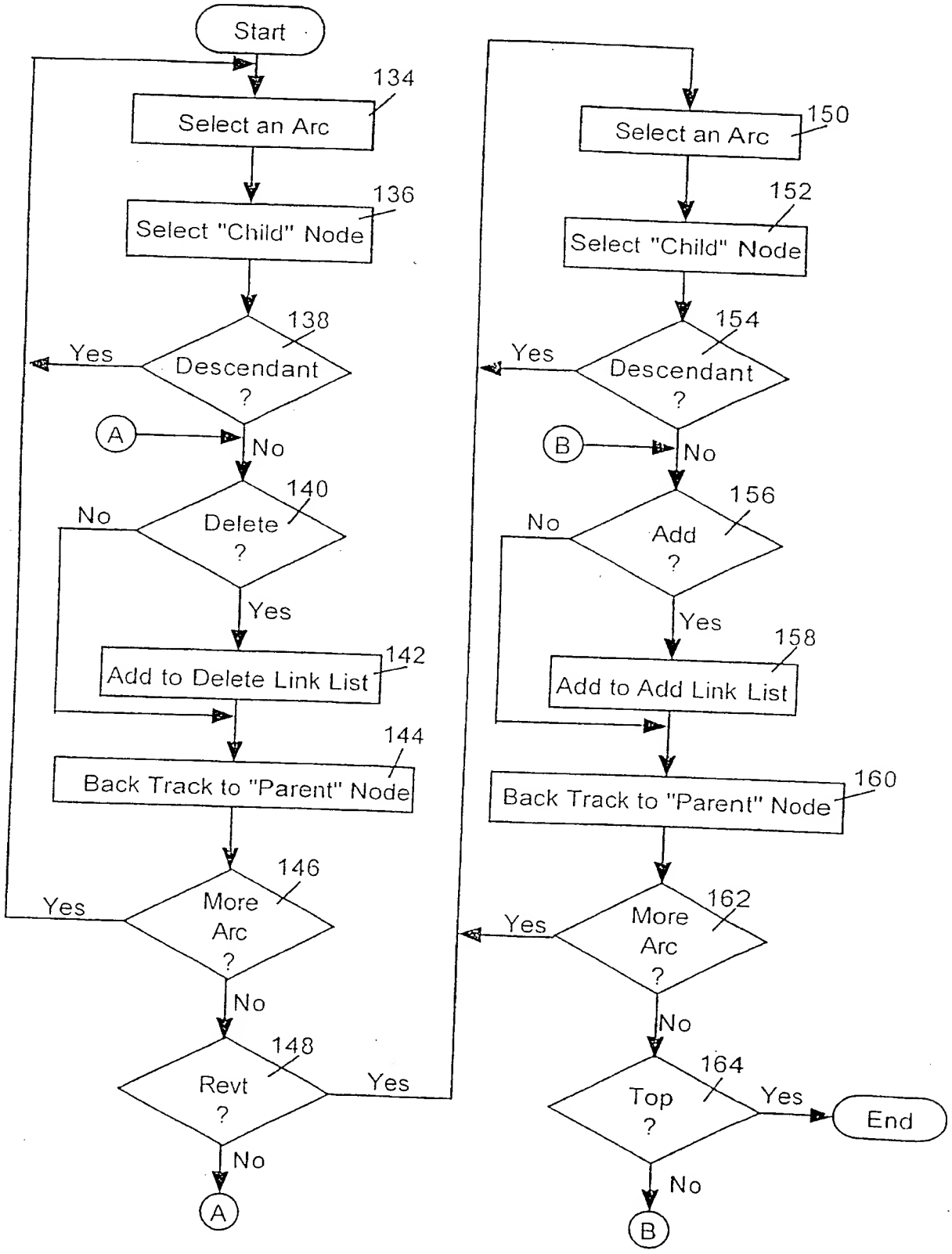

Root
- Set_Data
- Build_Ct_FDG
- Build_Rec_FDG
- Enumerate
- Output
18
10
Current Configuration
20
Target Configuration
22
Current FDG
24
Target FDG
26
Reconfiguration FDG
28
Reconfiguration Linklist
29
Reconfiguration Script
30

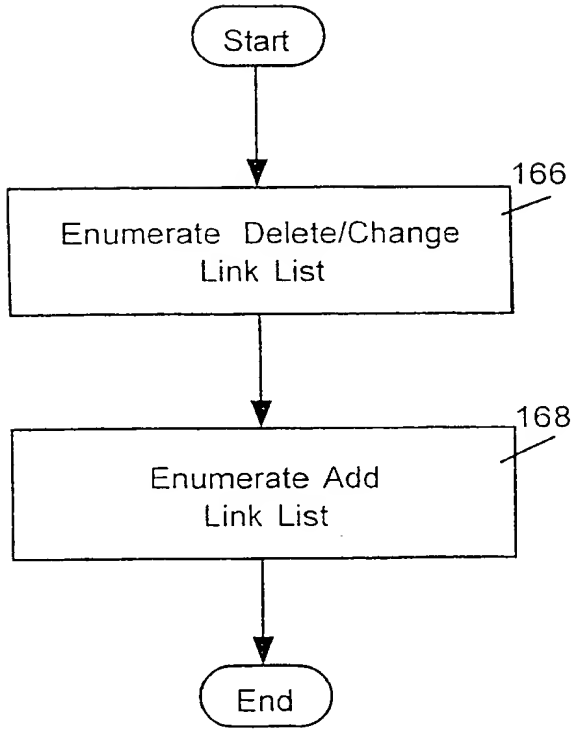

Current/Target
34
Station
Len, Sec Type, ID, Button List
36
32
Button
#, Feature List
38
Feature
ID, Parameter List
40
Parameter
ID, Parameter Value
42

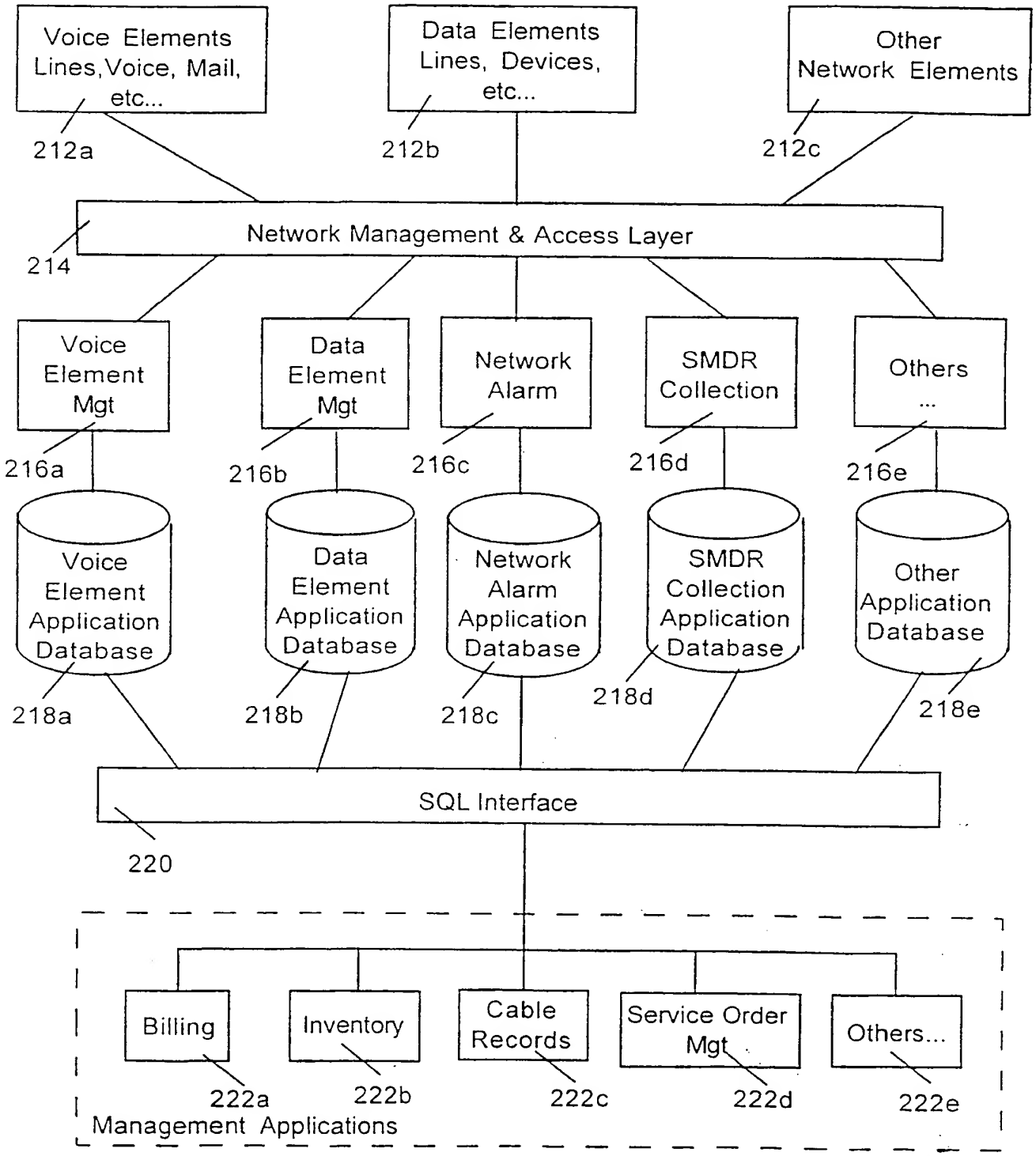

Start
Create Station Instance, Set Station Len, Set Type, ID, etc. To Input Values
102
Append Button ID to Button List, & Create a Button Instance
104
Append Feature ID to Button's Feature List, & Create a Feature Instance
106
Append Parameter to Feature's Parameter list, & Create a Parameter Instance
108
Set Parameter Values
110
More Parameters ?
112
Yes
No
More Features ?
114
Yes
No
More Buttons ?
116
Yes
No
More Stations ?
118
Yes
No
End

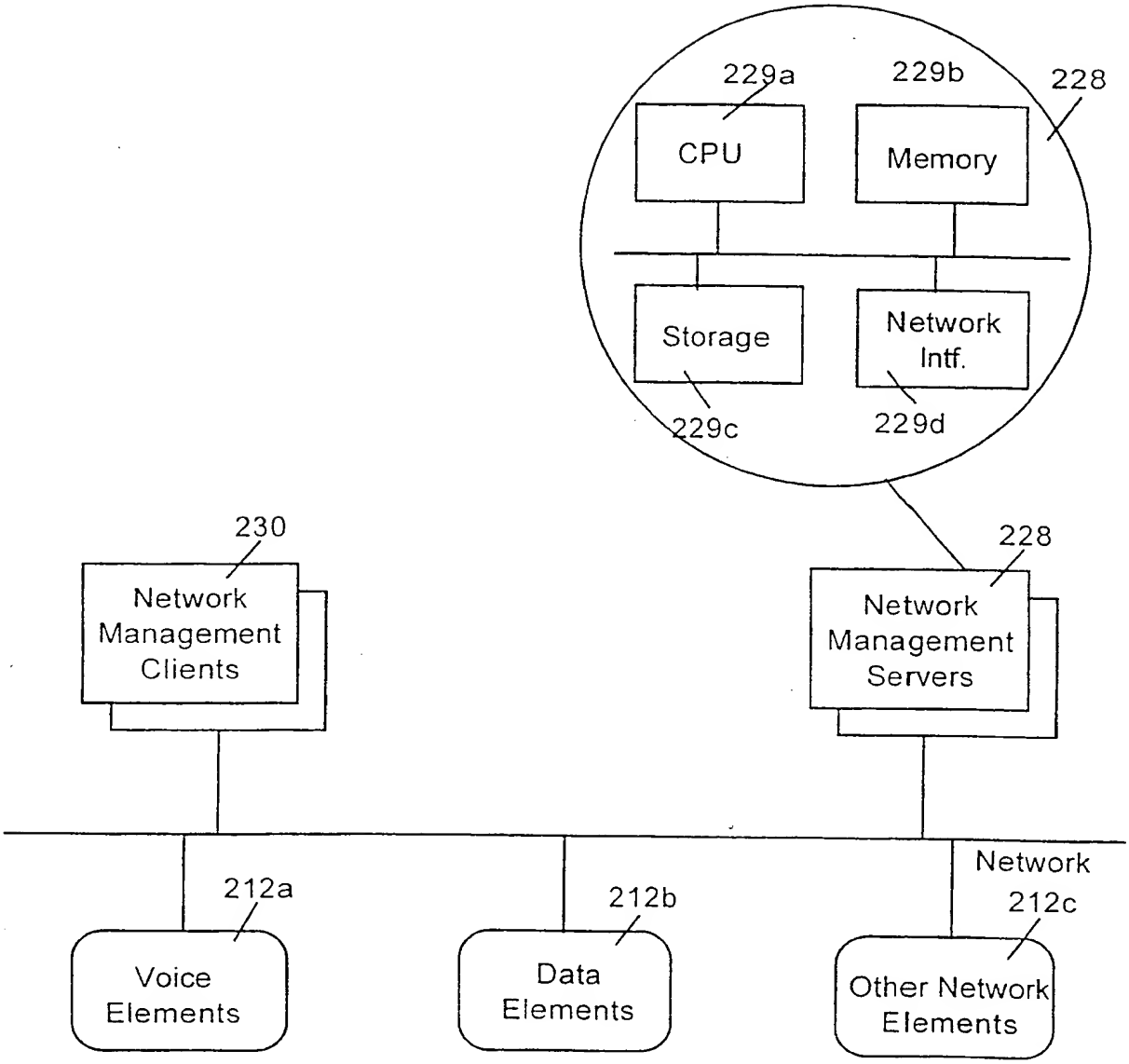

Start
For Each Button/Feature Create a Node
120
Retrieve Relationships For All Button/Features
122
For Each Required Scope, Create a Node
124
For Each Relationship, Create an Arc
126
End


Start
128
Indentify Kernel For Change FDG
130
Create "Delete" Nodes For Reconfigure FDG in View of Current Image
132
Create "Addition" Nodes For Reconfigure Change FDG in View of Target Image
End

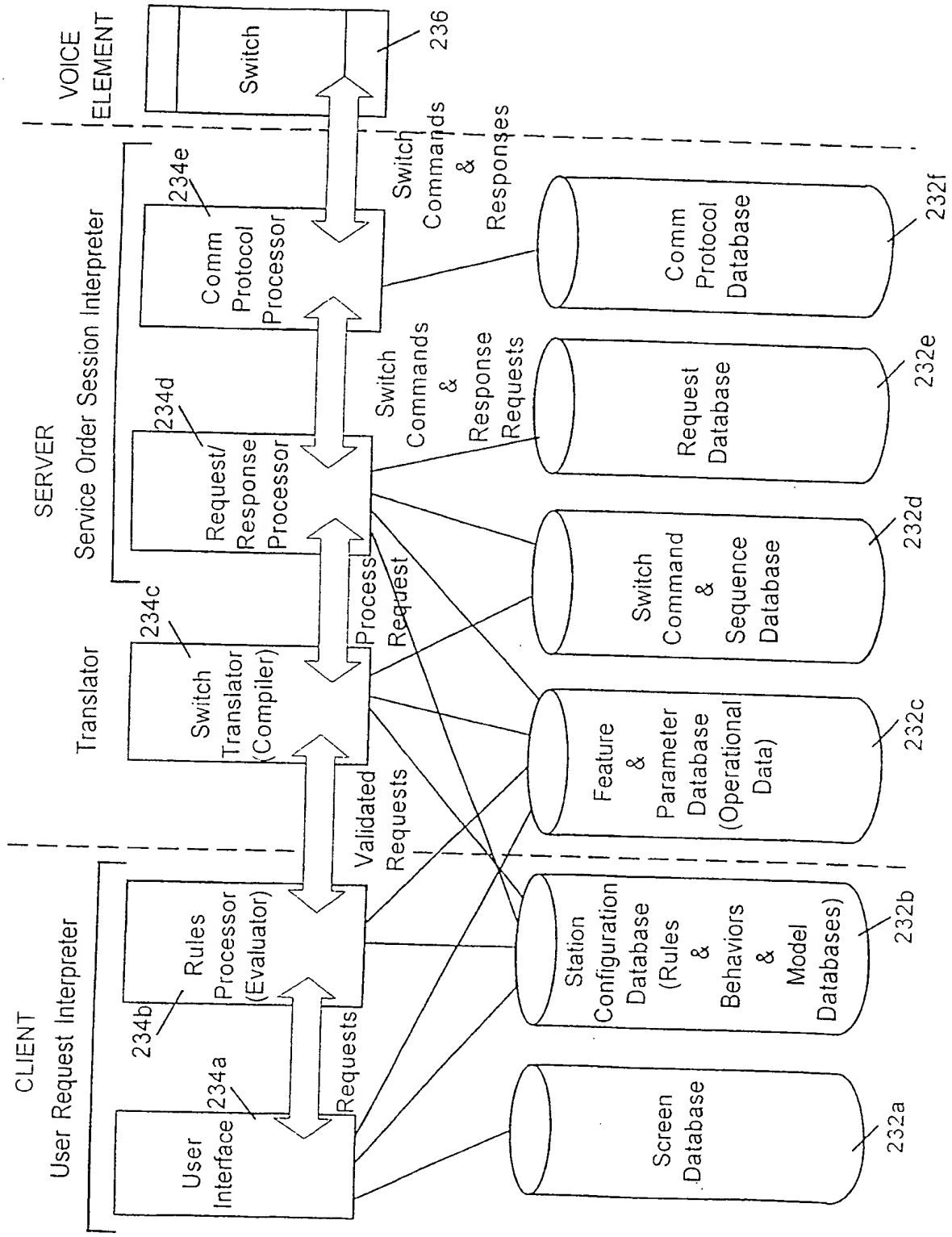

Start
134
Select an Arc
136
Select "Child" Node
138
Descendant ?
Yes
A
No
140
Delete ?
No
Yes
142
Add to Delete Link List
144
Back Track to "Parent" Node
146
More Arc ?
Yes
No
148
Revt ?
Yes
No
A
150
Select an Arc
152
Select "Child" Node
154
Descendant ?
Yes
B
No
156
Add ?
No
Yes
158
Add to Add Link List
160
Back Track to "Parent" Node
162
More Arc ?
Yes
No
164
Top ?
Yes
End
No
B


Start
166
Enumerate Delete/Change Link List
168
Enumerate Add Link List
End


Voice Elements Lines,Voice, Mail, etc...
212a
Data Elements Lines, Devices, etc...
212b
Other Network Elements
212c
Network Management & Access Layer
214
Voice Element Mgt
216a
Data Element Mgt
216b
Network Alarm
216c
SMDR Collection
216d
Others ...
216e
Voice Element Application Database
218a
Data Element Application Database
218b
Network Alarm Application Database
218c
SMDR Collection Application Database
218d
Other Application Database
218e
SQL Interface
220
Billing
222a
Inventory
222b
Cable Records
222c
Service Order Mgt
222d
Others...
222e
Management Applications

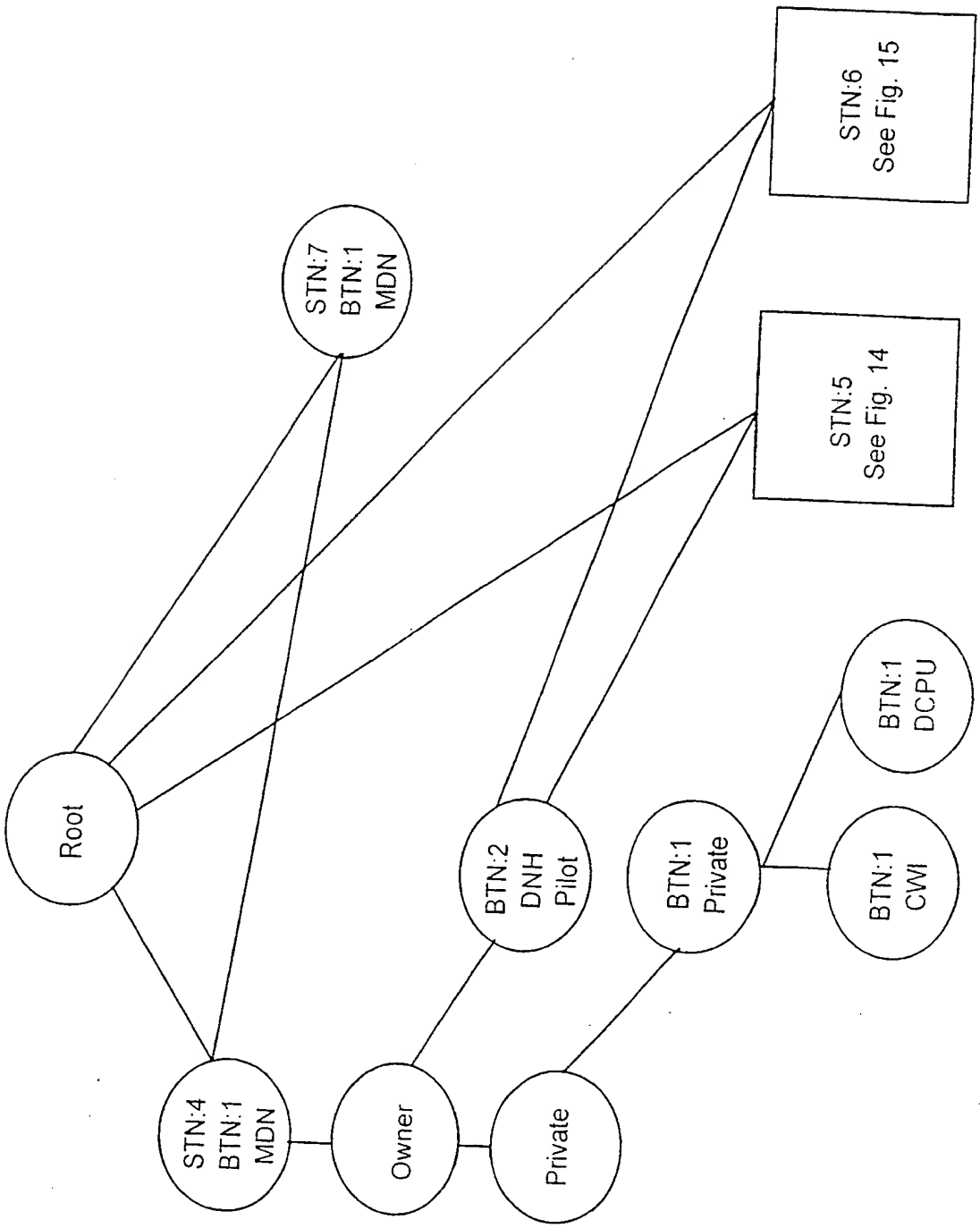

229a
229b
228
CPU
Memory
Storage
Network Intf.
229c
229d
230
228
Network Management Clients
Network Management Servers
Network
212a
212b
212c
Voice Elements
Data Elements
Other Network Elements

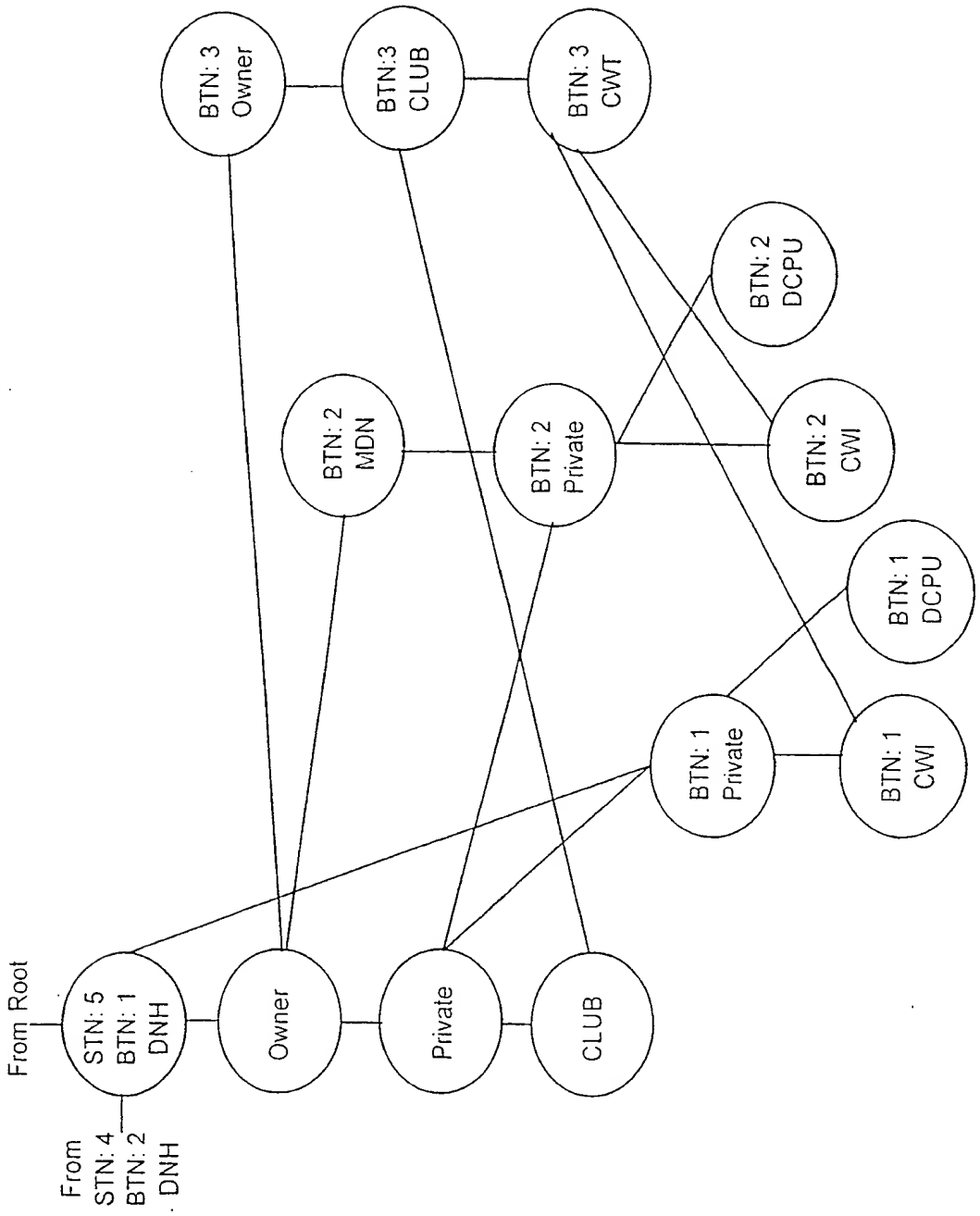
224

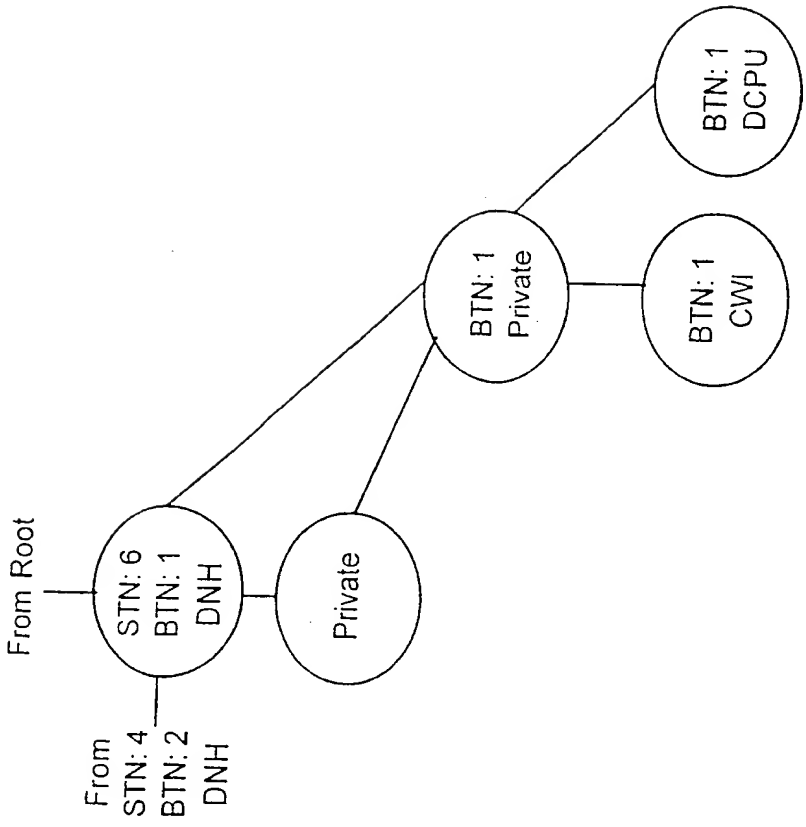

VOICE ELEMENT
Switch
236
Switch Commands & Responses
SERVER
Service Order Session Interpreter
234e
Comm Protocol Processor
Comm Protocol Database
232f
234d
Request/ Response Processor
Switch Commands & Response Requests
Request Database
232e
Translator
234c
Switch Translator (Compiler)
Process Request
Switch Command & Sequence Database
232d
Validated Requests
Feature & Parameter Database (Operational Data)
232c
CLIENT
User Request Interpreter
234b
Rules Processor (Evaluator)
Station Configuration Database (Rules & Behaviors & Model Databases)
232b
234a
User Interface
Requests
Screen Database
232a

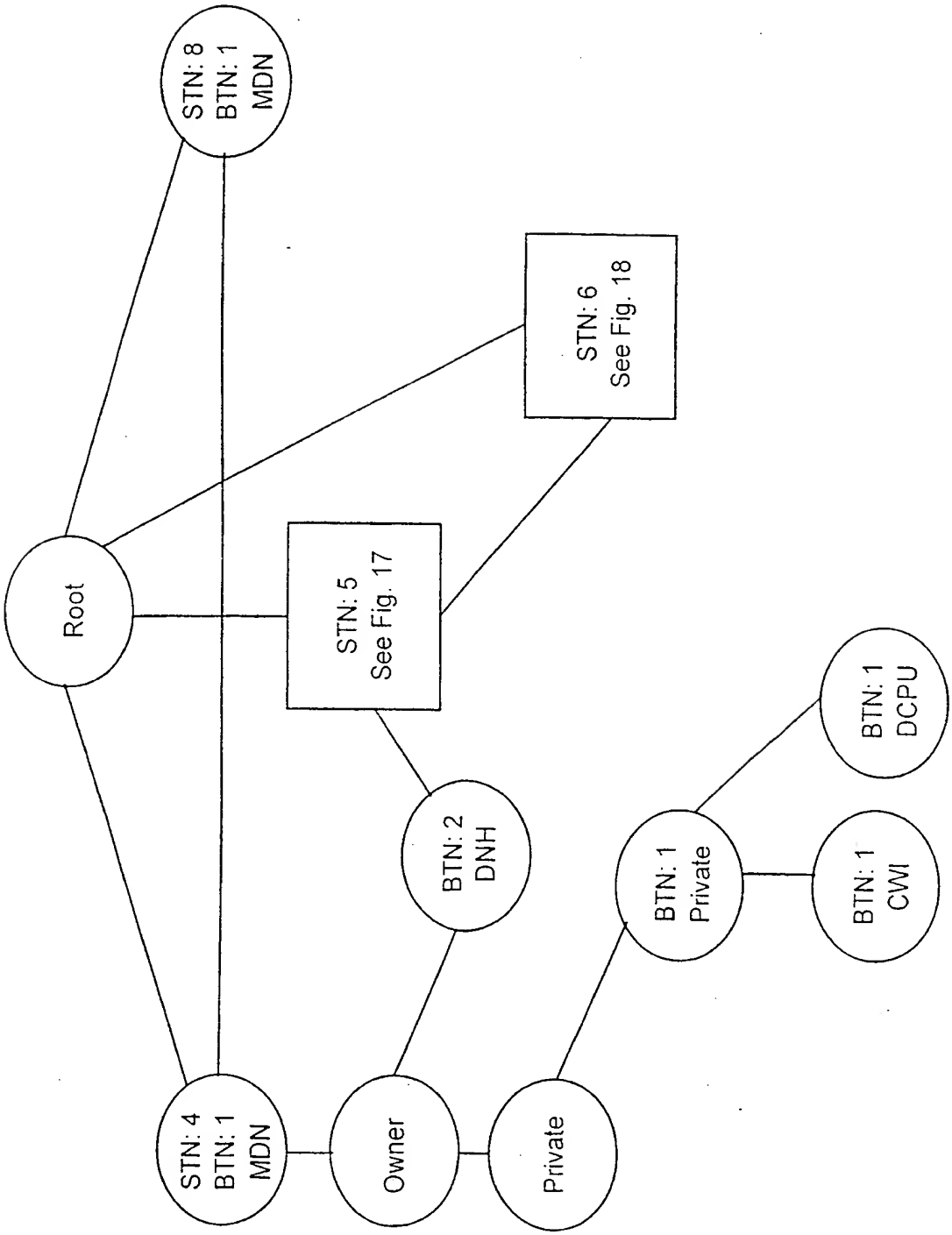
```
Current Imate {
    . . .
Station 4 {
        Len: "Host 0 2 1 11 4",
        SetType: M5312
            Button: 1 {
                Feature: DCPU
                Feature: CWI
                Feature: MDN {
                    MDNDN: 206-5101
                        . . . }}
            Button: 2 {
                Feature: DNH {
                    DNHGRID: 206-5118
                    DNHMEMNO: 0
                        . . . }}}
Station: 5 {
        Len: "Host 0 2 1 11 5",
        SetType: M5312
            Button: 1 {
                Feature: DCPU
                Feature: CWI
                Feature: DNH {
                    DNHGRID: 260-5118
                    DNHMEMNO: 1
                        . . .       }
            Button: 2 {
                Feature: DCPU
                Feature: CWI
                    . . .           }
            Button: 3 {
                Feature: CWT}}
Station: 6 {
        Len: "Host 0 2 1 11 6",
        SetType: BTS
            Button: 1 {
                Feature: DCPU
                Feature: CWI
                Feature: DNH {
                    DNHGRID: 206-5118
                    DNHMEMNO: 2
                        . . .       }}}
Station: 7 {
        Len: "Host 0 2 1 11 7",
        SetType: BTS
            Button: 1 {
                Feature: MDN {
                    MDNDN: 206-5101
                        . . . }}}}
```

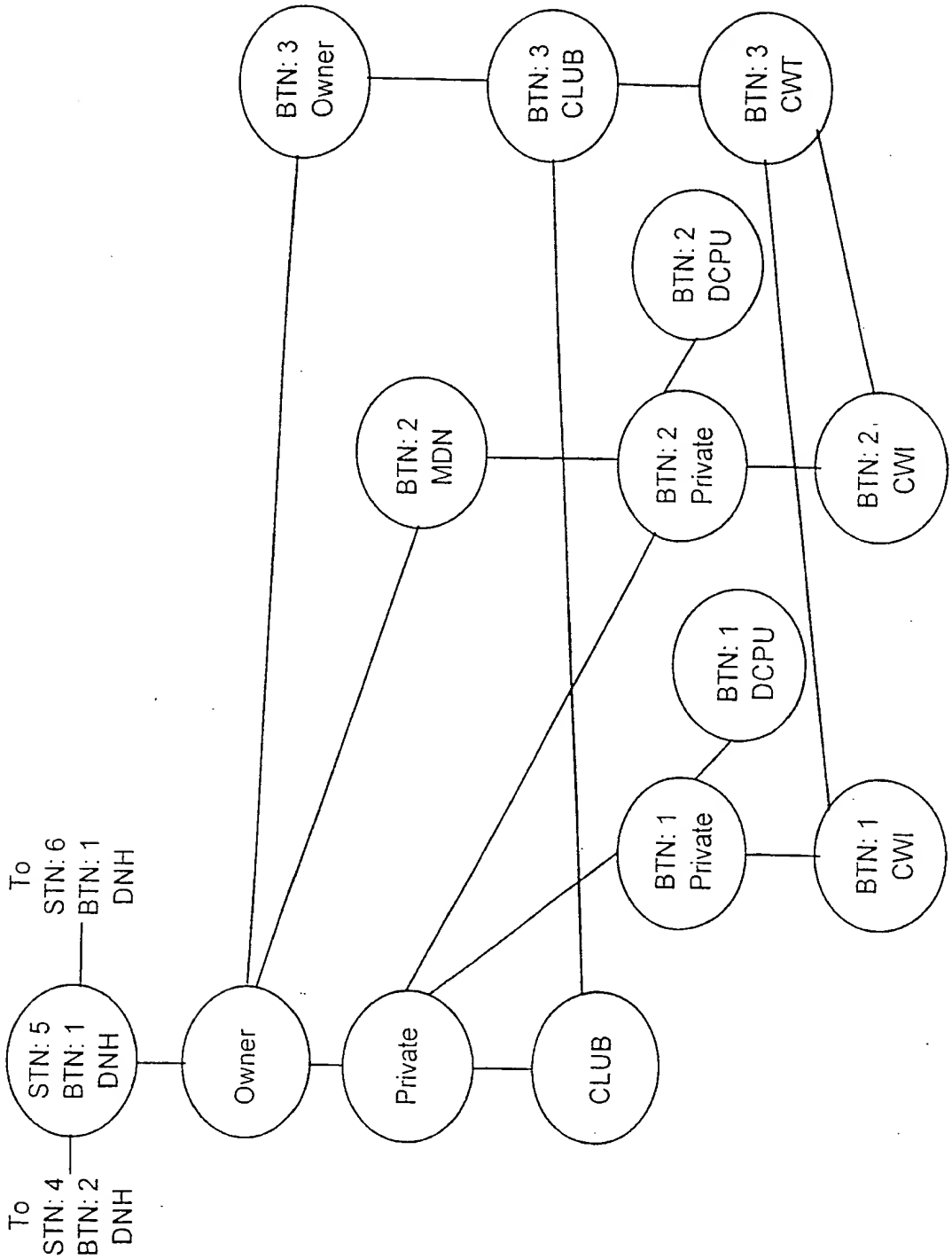
```
Target Imate {
    . . .
Station 4 {
        Len: "Host 0 2 1 11 4",
        SetType: M5312
            Button: 1 {
                Feature: DCPU
                Feature: CWI
                Feature: MDN {
                    MDNDN: 206-5101
                        . . . }}
            Button: 2 {
                Feature: DNH {
                    DNHGRID: 206-5118
                    DNHMEMNO: 1
                        . .     }}}
Station: 5 {
        Len: "Host 0 2 1 11 5",
        SetType: M5312
            Button: 1 {
                Feature: DCPU
                Feature: CWI
                Feature: DNH {
                    DNHGRID: 260-5118
                    DNHMEMNO: 0
                        . . .   }
            Button: 2 {
                Feature: DCPU
                Feature: CWI
                    . . .       }
            Button: 3 {
                Feature: CWT}}
Station: 6 {
        Len: "Host 0 2 1 11 6",
        SetType: BTS
            Button: 1 {
                Feature: DCPU
                Feature: CWI
                Feature: DNH {
                    DNHGRID: 206-5118
                    DNHMEMNO: 2
                        . . .   }}}
Station: 8 {
        Len: "Host 0 2 1 11 8",
        SetType: BTS
            Button: 1 {
                Feature: MDN {
                    MDNDN: 206-5101
                        . . .   }}}}
```

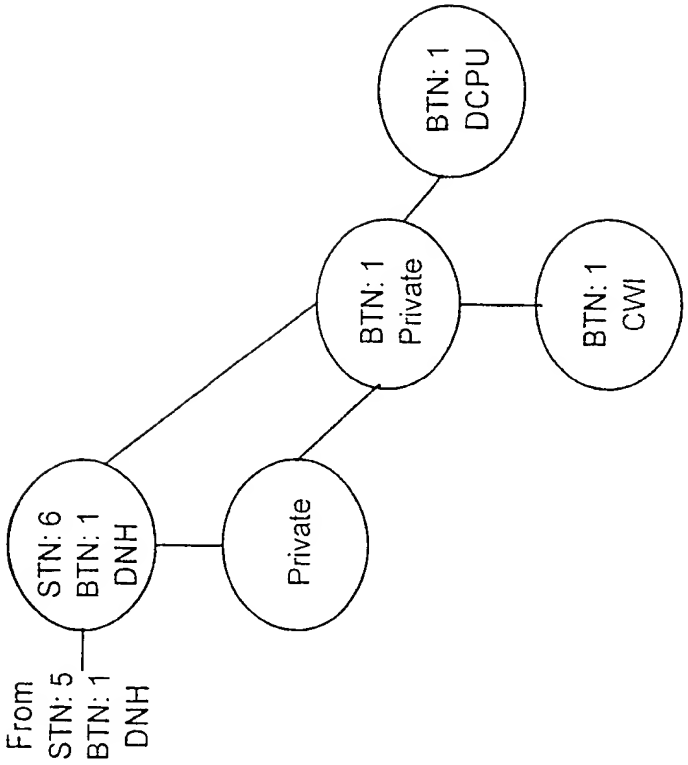

STN:6
See Fig. 15
STN:7
BTN:1
MDN
STN:5
See Fig. 14
Root
BTN:1
DCPU
BTN:2
DNH
Pilot
BTN:1
Private
BTN:1
CWI
STN:4
BTN:1
MDN
Owner
Private

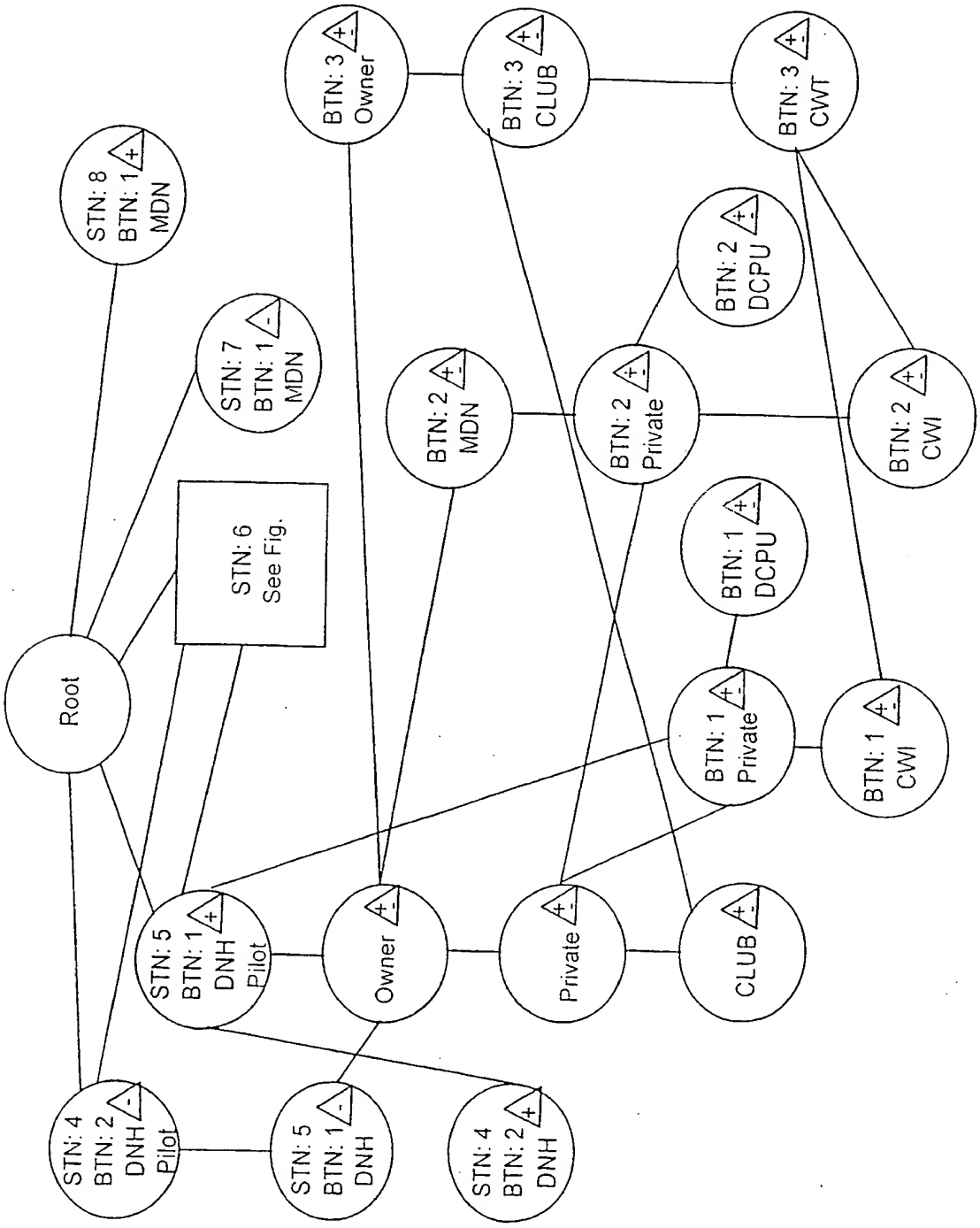

From Root
From STN: 4 BTN: 2 DNH
STN: 5 BTN: 1 DNH
Owner
Private
CLUB
BTN: 3 Owner
BTN:3 CLUB
BTN: 3 CWT
BTN: 2 MDN
BTN: 2 Private
BTN: 2 DCPU
BTN: 2 CWI
BTN: 1 Private
BTN: 1 DCPU
BTN: 1 CWI

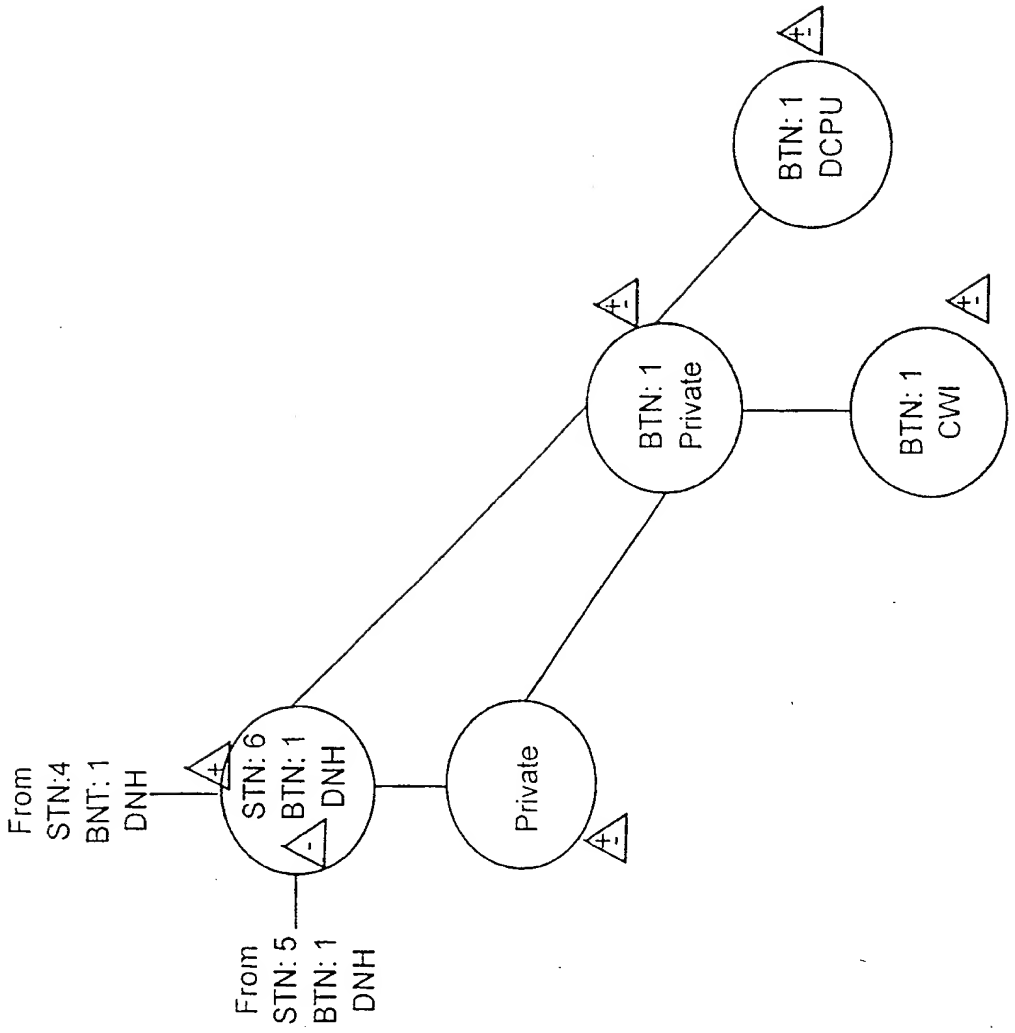

From Root
STN: 6
BTN: 1
DNH
From
STN: 4
BTN: 2
DNH
Private
BTN: 1
Private
BTN: 1
CWI
BTN: 1
DCPU

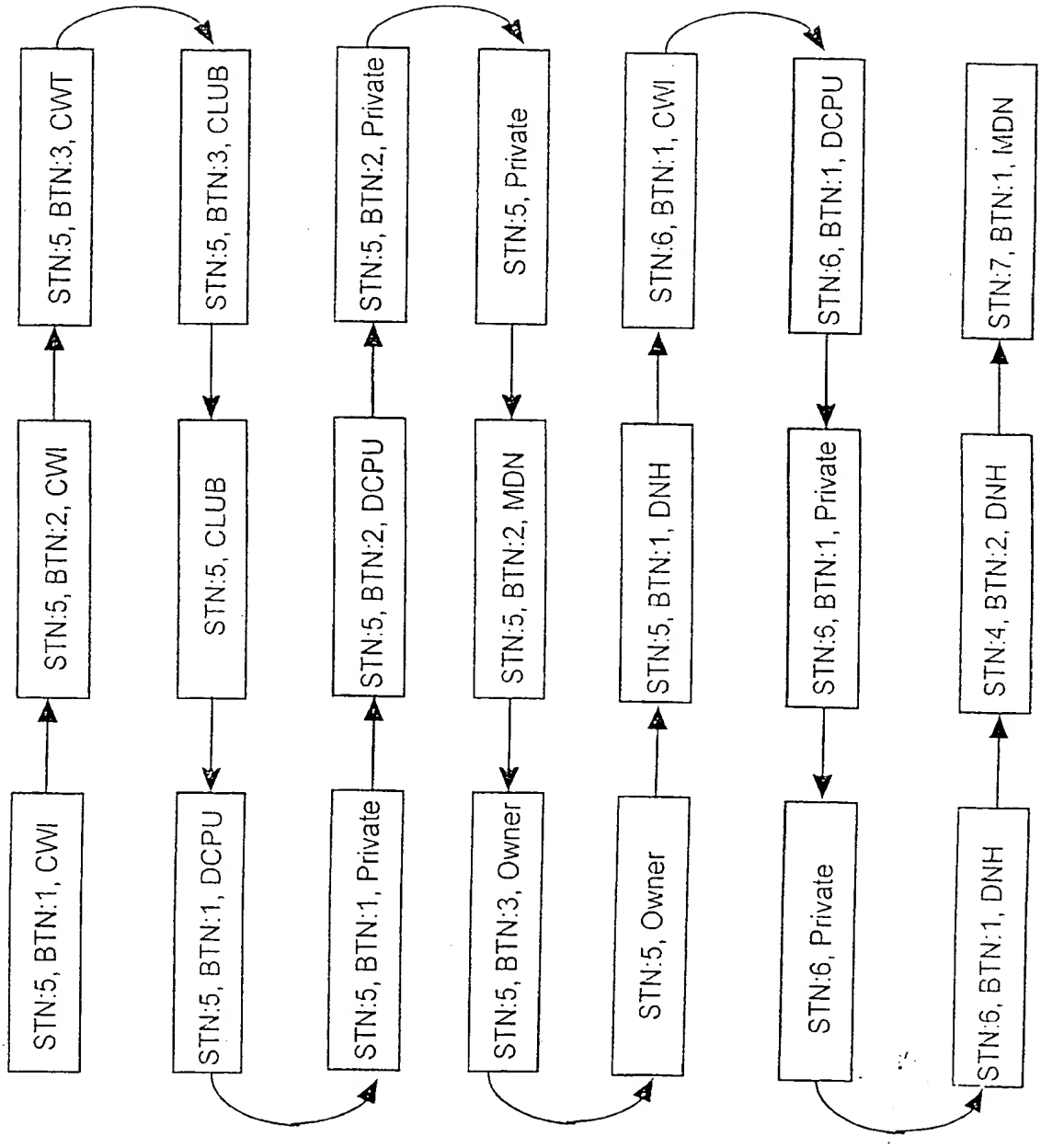

STN: 8
BTN: 1
MDN
STN: 6
See Fig. 18
Root
STN: 5
See Fig. 17
BTN: 1
DCPU
BTN: 2
DNH
BTN: 1
Private
BTN: 1
CWI
STN: 4
BTN: 1
MDN
Owner
Private

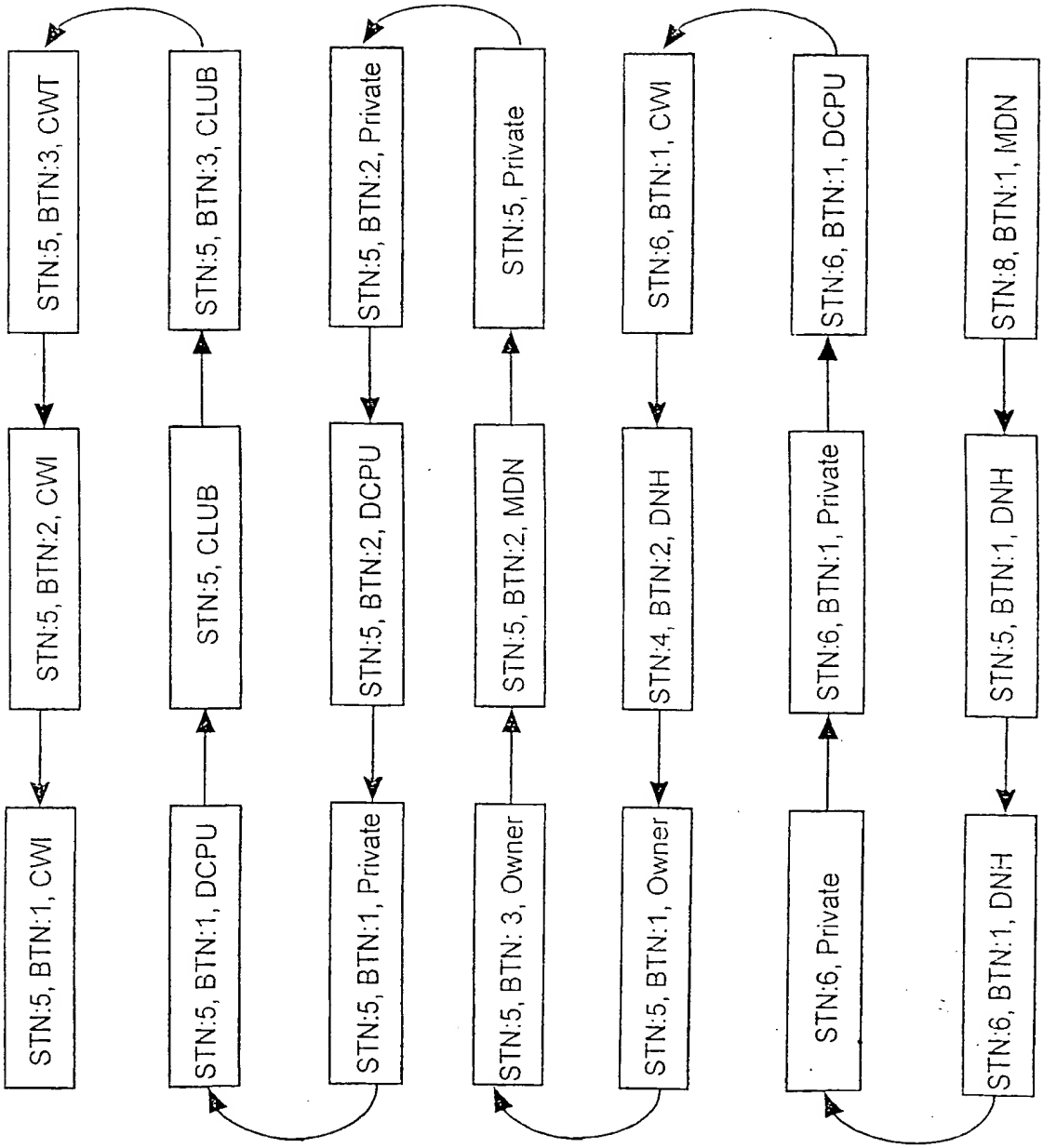

To STN: 4 BTN: 2 DNH
STN: 5 BTN: 1 DNH
To STN: 6 BTN: 1 DNH
Owner
Private
CLUB
BTN: 3 Owner
BTN: 3 CLUB
BTN: 3 CWT
BTN: 2 DCPU
BTN: 2 MDN
BTN: 2 Private
BTN: 2. CWI
BTN: 1 DCPU
BTN: 1 Private
BTN: 1 CWI


BTN: 1
DCPU
BTN: 1
Private
BTN: 1
CWI
STN: 6
BTN: 1
DNH
Private
From
STN: 5
BTN: 1
DNH

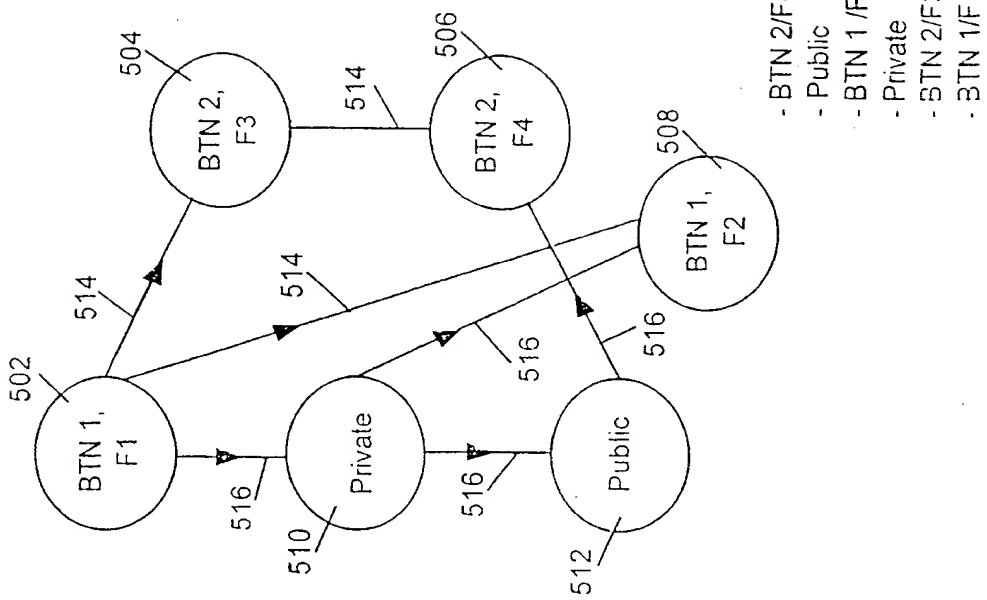

Root
STN: 8 BTN: 1 MDN
STN: 7 BTN: 1 MDN
STN: 6 See Fig.
STN: 5 BTN: 1 DNH Pilot
STN: 4 BTN: 2 DNH Pilot
STN: 5 BTN: 1 DNH
STN: 4 BTN: 2 DNH
BTN: 3 Owner
BTN: 3 CLUB
BTN: 3 CWT
BTN: 2 DCPU
BTN: 2 MDN
BTN: 2 Private
BTN: 2 CWI
BTN: 1 DCPU
BTN: 1 Private
BTN: 1 CWI
Owner
Private
CLUB

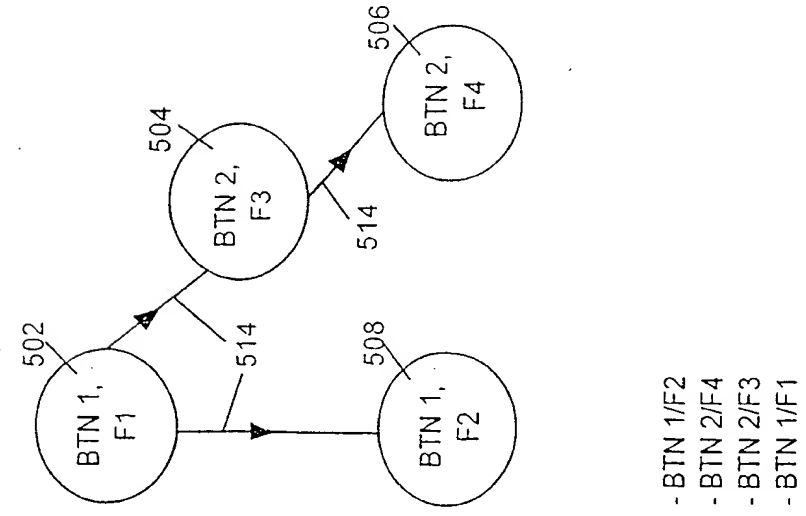

From
STN:4
BNT: 1
DNH
STN: 6
BTN: 1
DNH
From
STN: 5
BTN: 1
DNH
Private
BTN: 1
Private
BTN: 1
CWI
BTN: 1
DCPU

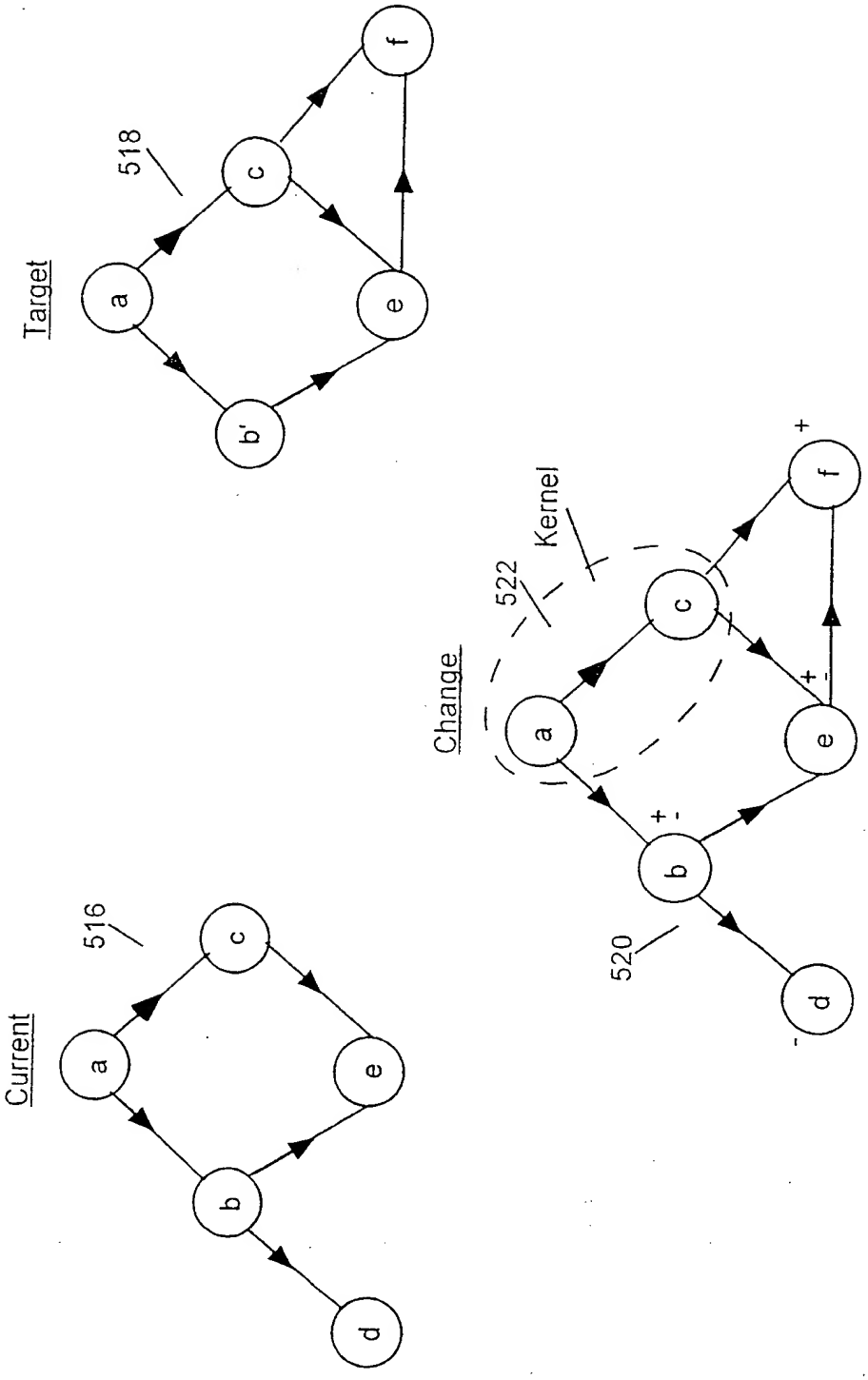

STN:5, BTN:1, CWI
STN:5, BTN:2, CWI
STN:5, BTN:3, CWT
STN:5, BTN:1, DCPU
STN:5, CLUB
STN:5, BTN:3, CLUB
STN:5, BTN:1, Private
STN:5, BTN:2, DCPU
STN:5, BTN:2, Private
STN:5, BTN:3, Owner
STN:5, BTN:2, MDN
STN:5, Private
STN:5, Owner
STN:5, BTN:1, DNH
STN:6, BTN:1, CWI
STN:6, Private
STN:6, BTN:1, Private
STN:6, BTN:1, DCPU
STN:6, BTN:1, DNH
STN:4, BTN:2, DNH
STN:7, BTN:1, MDN


STN:5, BTN:3, CWT
STN:5, BTN:2, CWI
STN:5, BTN:1, CWI
STN:5, BTN:3, CLUB
STN:5, CLUB
STN:5, BTN:1, DCPU
STN:5, BTN:2, Private
STN:5, BTN:2, DCPU
STN:5, BTN:1, Private
STN:5, Private
STN:5, BTN:2, MDN
STN:5, BTN: 3, Owner
STN:6, BTN:1, CWI
STN:4, BTN:2, DNH
STN:5, BTN:1, Owner
STN:6, BTN:1, DCPU
STN:6, BTN:1, Private
STN:6, Private
STN:8, BTN:1, MDN
STN:5, BTN:1, DNH
STN:6, BTN:1, DNH

```
- $ HOST 0 2 1 11 5 1 CWI $
- $ HOST 0 2 1 11 5 2 CWI $
- $ HOST 0 2 1 11 5 3 CWT $
//rmv scope node: 3/Club
//rmv scope node: Club
- $ HOST 0 2 1 11 5 1 DCPU $
//rmv scope node: 1 /Private
- $ HOST 0 2 1 11 5 2 DCPU $
//rmv scope node: 2/Private
//rmv scope node:  Private
- $ HOST 0 2 1 11 5 2 MDN $
//rmv scope node: 3/Owner
//rmv scope node: Owner
- $ HOST 0 2 1 11 5 1 DNH $
- $ HOST 0 2 1 11 6 1  DCPU $
- $ HOST 0 2 1 11 6 1 CWI $
//rmv scope node: 1/Private
//rmv scope node: Private
- $ HOST 0 2 1 11 6 1 DNH $
- $ HOST 0 2 1 11 4 2 DNH $
- $ HOST 0 2 1 11 7 1 MDN $
+ $ HOST 0 2 1 11 5 1 DNH $
+ $ HOST 0 2 1 11 6 1 DNH $
//add scope node: Private
//add scope node: 1/Private
+ $ HOST 0 2 1 11 6 1 CWI $
+ $ HOST 0 2 1 11 6 1 DCPU $
+ $ HOST 0 2 1 11 4 2 DNH $
//add scope node: Owner
//add scope node: 3/Owner
+ $ HOST 0 2 1 11 5 2 MDN $
//add scope node: Private
//add scope node: 2/Private
+ $ HOST 0 2 1 11 5 2 DCPU $
//add scope node : 1/Private
+ $ HOST 0 2 1 11 5 1 DCPU $
//add scope node: Club
//add scope node: 3/Club
+ $ HOST 0 2 1 11 5 3 CWT $
+ $ HOST 0 2 1 11 5 2 CWI $
+ $ HOST 0 2 1 11 5 1 CWI $
+ $ HOST 0 2 1 11 8 1 MDN $
```


BTN 1, F1
BTN 2, F3
BTN 2, F4
BTN 1, F2
Private
Public
502
504
506
508
510
512
514
516
- BTN 2/F4
- Public
- BTN 1 /F2
- Private
- BTN 2/F3
- BTN 1/F1


BTN 1, F1
BTN 2, F3
BTN 2, F4
BTN 1, F2
502
504
506
508
514
- BTN 1/F2
- BTN 2/F4
- BTN 2/F3
- BTN 1/F1


Target
518
a
b'
c
e
f
Current
516
a
b
c
d
e
Change
Kernel
522
520
a
b
c
d
e
f
+
-

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :		(11) International Publication Number:
H04J 3/06	**A3**	**WO 98/34350**
		(43) International Publication Date: 6 August 1998 (06.08.98)

(21) International Application Number: PCT/US98/01729

(22) International Filing Date: 29 January 1998 (29.01.98)

(30) Priority Data:
08/795,917 5 February 1997 (05.02.97) US

(71) Applicant: FIRSTTEL SYSTEMS CORPORATION [US/US]; 100 Marine World Parkway, Redwood Shores, CA 94065 (US).

(72) Inventors: MA, Xiwen; 1615 Chestnut Street, Berkeley, CA 94702 (US). TIAN, XiaoLin; 10192 Park Circle West #2, Cupertino, CA 95014 (US). BERRIATUA, Steve; 778 Olive Court, San Bruno, CA 94066 (US). MOLONEY, Simon; 150 D Street, Redwood City, CA 94063 (US). RUSSELL, Mark; 34 Sonoma Street, San Francisco, CA 94123 (US).

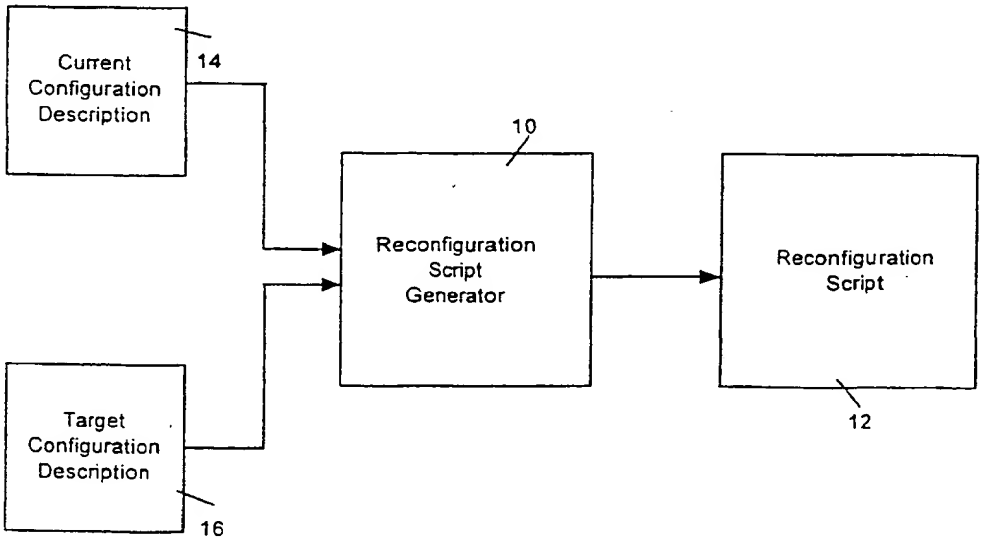
(74) Agents: AU YEUNG, Aloysius, T., C. et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 7th floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025 (US).

(81) Designated States: AL, AM, AT, AT (Utility model), AU (Petty patent), AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, CZ (Utility model), DE, DE (Utility model), DK, DK (Utility model), EE, EE (Utility model), ES, FI, FI (Utility model), GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SK (Utility model), SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(88) Date of publication of the international search report:
17 December 1998 (17.12.98)

(54) Title: AUTOMATIC GENERATION OF RECONFIGURATION SCRIPTS FOR TELECOMMUNICATION DEVICES



(57) Abstract

An apparatus is programmed with a plurality of programming instructions for automatically generating a reconfiguration script (12) comprising a plurality of configuration commands for reconfiguring a plurality of telecommunication devices of a telecommunication system, based on a current (14) and a target (16) descriptive image of the telecommunication system. The current descriptive image specifies the devices and their features included in the current configuration of the telecommunication system, whereas the target image specifies the devices and their features to be included in the target configuration of the communication system. The plurality of programming instructions (10) generate the reconfiguration script (12) employing feature dependency graph (FDG) data structures, a device model modeling the rules and behaviors of the telecommunication devices, and feature deletion/addition linklists.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/01729

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :H04J 3/06
US CL : 455/560; 455/439
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 455/560; 455/439; 395/500

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

IEEE, APS

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5,450,486 A (MAAS ET AL.) 12 September 1995, col. 4-7	1-26
A,P	US 5,653,559 A (STIEB ET AL.) 05 August 1997, col. 2-4	1-26
A,P	US 5,664,008 A (BOSSI ET AL.) 02 September 1997, col. 3-11	1-26
A,E	US 5,754,628 A (BOSSI ET AL.) 19 May 1998, col. 3-11	1-26
A,E	US 5,771,386 A (BAUMBAUER) 23 June 1998, col. 3-8	1-26

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search
23 SEPTEMBER 1998

Date of mailing of the international search report
29 OCT 1998

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. (703) 305-3230

Authorized officer
KEVIN TESKA
Telephone No. (703) 305-9704

Form PCT/ISA/210 (second sheet)(July 1992)*